Adapting Bayesian Networks to Predict Complex Systems using Small Datasets

Anastacia MacAllister, Eliot Winer
Iowa State University
Ames, IA
anastac@iastate.edu, ewiner@iastate.edu

ABSTRACT

Increasingly commercial companies including Google, Amazon and Apple are using machine learning (ML) to predict customer behaviour and market trends. As these ML methods mature, they will continue to help improve commercial sector decision making, and potentially military processes as well. Reports suggest that the DoD alone could save $32 billion a year by increasing logistics and operational efficiency, savings that ML could help facilitate. Unfortunately, many ML methods require millions of known data points to train a system before its predictive capabilities can be realized. However, for many military processes, only relatively small data sets are available (i.e. hundreds to thousands of points). This paper explores a specific ML method, Bayesian Networks (BN), to function on problems with small amounts of known data. Specifically, this work investigates the feasibility of using Kriging and Radial Basis Functions to augment existing data available for training BNs. In addition, tuning BN parameters to increase network accuracy using Particle Swarm Optimization is also presented. Combined results from three different datasets suggest that pairing data generation and prior probability approximation can allow BNs to more accurately predict a system's outcome with small amounts of known data, potentially up to 80% or higher. Ultimately, as strategies outlined in the paper continue to develop they could help aid the implementation of BNs for a wide range of military processes. This would allow inefficiencies to be predicted before actual time, materials, and person hours are wasted.

ABOUT THE AUTHORS

Anastacia MacAllister is a graduate student in Mechanical Engineering and Human-Computer Interaction with a Minor in Computer Science at Iowa State University's Virtual Reality Applications Center. She is working on developing Augmented Reality work instructions for complex assembly, intelligent team tutoring systems, and machine learning techniques for small data sets.

Eliot Winer, Ph.D., is an associate director of the Virtual Reality Applications Center and professor of Mechanical Engineering at Iowa State University. He has led numerous projects for the D0D including development of a next-generation mixed-reality virtual and constructive training environment for the U.S. Army. Dr. Winer has over 18 years of experience working in virtual reality and 3D computer graphics technologies on sponsored projects for the Department of Defense, Air Force Office of Scientific Research, Department of the Army, National Science Foundation, Department of Agriculture, Boeing, and John Deere.

Adapting Bayesian Networks to Predict Complex Systems using Small Datasets

Anastacia MacAllister, Eliot Winer
Iowa State University
Ames, IA
anastac@iastate.edu, ewiner@iastate.edu

INTRODUCION

With the increasing prevalence of commodity sensors and computing devices, data can now be collected for relatively inconsequential costs and effort. Using increasingly affordable commodity sensors, collecting data on a manufacturing process, customer satisfaction, or disaster response can now lead to a wealth of information, information that companies are increasingly reluctant to ignore in the digital age (Malinova & Mendling, 2015). With the rise of the digital economy, data is being compared to oil as an economic engine and as a result becoming a fiercely guarded trade secret (Rotella, 2012). With the new focus on data, many organizations, like Amazon, Google, and Microsoft, are recognizing how it can give them a competitive edge to help improve their product offerings (Biewald, 2016).

While raw data collection is as simple as ever, distilling actionable strategies from it is extremely challenging. This is due to the magnitude of information collected on a daily basis that must be interpreted. In order to make use of all that information to guide important decisions, strategies need to be employed to sort through noise and produce something is actionable (Holzinger, Dehmer, & Jurisica, 2014). This sorting or mining through data is often accomplished by utilizing machine learning techniques that can make accurate predictions and forecasts based on trends. Not only can these machine learning tools aid in decision making, they can also be used to understand how different decisions or events impact a system as a whole. This ability to make sense of complex situations with numerous variables is invaluable in today's world where companies and governments are operating on increasingly tight budgets.

While using sensors to collect data is becoming more cost efficient, in some real-world cases one cannot often collect enough data points to use popular machine learning techniques like Neural Networks or Bayesian Networks, which can require millions of data points. In some domains, like manufacturing, battlefield training, or medical procedures, events of interest may only happen a handful of times throughout the year. As a result, collecting thousands to millions of unique data points is not possible in a reasonable amount of time. One specific example highlighting this data volume issue is aircraft manufacturing. The process is very complex and involves many different collaborators from union labor to dozens of suppliers all impacting the finished product. Specifically, worker suitability can significantly impact assembly and manufacturing process outcomes. The ability to assign the correct worker to a job could provide a competitive edge by making sure their skills are suitably matched with a task (Ong, Ato, Umar, & Oshino, 2016). However, there might only be a relatively small number of planes produced every month (i.e. 10-30), meaning there is not enough worker data to construct a model to predict competencies (BBC, 2015) until many years have passed. The lack of data means there is not a way to use machine learning to accurately understand the relationships between variables and to predict how changes might impact the production process. As a result, these types of complex processes cannot use the powerful predictive analytics of machine learning without strategies that augment the limited quantities of data available.

If smaller data sets could be augmented to a point where they are compatible with machine learning approaches, a wider range of applications could take advantage of these predictive tools. In order to accomplish this, however, a machine learning approach needs to be selected that has enough transparency so one can understand potential break downs in the training process, allowing improvements to be made. Since each machine learning algorithm has unique characteristics, it is important to select a method that is transparent enough for this task (Loyer, Henriques, Fontul, & Wiseall, 2016). Approaches like linear classification are simple and widely used for less demanding machine learning applications (Loh, 2014). However, assumptions of constant distributions, and the requirement that terms only be combined in a linear manner, results in a method not well suited to complex processes. Decision trees are another popular machine learning approach in many domains (Rokach & Maimon, 2015). They are easy to understand graph

like structures, lending themselves well to applications in medicine (Podgorelec, Kokol, & Rozman, 2002). However, decision trees suffer from overfitting and issues handling noisy incomplete data, which are often found in many real-world applications (Bishop, 2006). Support vector machines (SVMs) are another widely popular machine learning approach. They are inherently a binary classification method that projects complex multivariate problems into an n-dimensional space using a kernel (Chen & Wang, 2007). While SVMs are a popular and powerful tool, using a n-dimensional kernel can be complex and challenging. This challenge does not allow for investigation of the small data set issue easily. Neural networks are one of the most powerful machine learning tools (Bernard, Chang, Popescu, & Graf, 2015). While they are a very powerful tool, they are very challenging to fully understand and dissect when trained. As a result, they are not a good option for investigating a machine learning algorithm for small data sets. Bayesian Networks (BN) are another widely used machine learning approach (Ropero, Flores, Rumí, & Aguilera, 2016). They combine expert knowledge in the form of a network structure and prior probability distribution with Bayesian Statistics. The easily understandable network structure paired with flexible Bayesian Statistical methods lends itself well to exploratory investigation of behaviors associated with small data sets for machine learning. As a result, Bayesian Networks (BN) were selected as the machine learning method for this work.

In addition to selecting a machine learning method for the work, a method of data augmentation is also required to train an accurate BN from a small sample size. It is important to select augmentation method that can capture the behavior of the dependent variable relationships in small data sets. This can facilitate data generation using a behavior model, instead of having to rely solely on what little data was collected. Theoretically, this could allow gaps in data collection to be filled. However, up to this point little work has been done looking augmentation strategies for ML using small datasets from real world processes. As such, little is known about what methods should be used to generate data for small dataset ML applications. Since different methods of data generation can produce very different results, any selected methods need to be tested and compared for accuracy (Krishnamurthy, 2005; Rusu & Rusu, 2006). This paper begins to explore the feasibility of using Kriging and Radial Basis Function models to generate data for training a Bayesian Network using four different network structures and three different datasets. It also explores the feasibility of using Particle Swarm Optimization (PSO) to further increase network accuracy by intelligently setting the priors of a Bayesian Network.

BACKGROUND

Statistical methods have long been used to help make sense of data and predict the likelihood of an event when provided with certain parameters. Statistical theory is used in areas spanning from reliability analysis to scheduling airline flights (Jacobs et al., 2012; Muller, 2003). The reason statistical methods are used increasingly, especially today, is their ability to suggest courses of action based on previously collected data. These suggestions benefit from the ability to look at far more relationships between variables than humans can and provide decision-making aids that are more unbiased (De Martino, Kumaran, Seymour, & Dolan, 2009; Hastie, 2001).

Bayesian Statistics

Powerful machine learning techniques using Bayesian Networks are made possible due to the resurgence of Bayesian statistical methods (Pearl, 1988). Bayes Theorem, shown in Equation 1, is unique from traditional statistical methods because it allows the incorporation of background information called the prior probability (Bayes, 1763). The term $P(A|B)$, called the posterior probability, represents the probability of A given that B occurs. The goal is to use what is known to calculate this unknown value. To calculate it involves using what is known on the right side of the equation. The term $P(A|B)$, is the likelihood of event B when A has already occurred in the population sample data. The term $P(A)$, is the prior probability of the event A happening anywhere in the population sample. The denominator term $P(B)$ is the probability of B occurring anywhere in the population sample data, which can be when A is either observed or not observed (Bayes, 1763).

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

The inclusion of a prior differs from traditional statistical likelihood-based approaches. These traditional approaches mainly estimate probabilities of events based on the observed sample population data (Orloff & Bloom, 2014). The introduction of the prior probability term allows for a correction or smoothing of the observed data described by the likelihood. Often this prior probability is thought of as an expert specified term. The combination of the prior and the likelihood give the posterior probability of an event occurring. Using Bayes Theorem in Equation 1, the two distributions can be combined to provide a best estimate of the probability of an event occurring.

Bayesian Networks

Conceptually, Bayes' Theorem, is straightforward when there are few events and few variables. In a simple problem, to calculate the probability of an event, multiply the prior by the likelihood. However, determining classification is challenging when there are multiple events with multiple variables for each event. Representing the relationships between variables can quickly become very complex. Helping to alleviate this problem, Bayesian Networks allow for the representation of dependencies and relationships between variables using Directed Acyclic Graphs (DAGs) (Nielsen & Jensen, 2007; Stephenson, 2000). These graphs are made up of vertices and edges. Vertices, also known as nodes, represent the variables that make up data points. Edges denote the causal relationships between the vertices. These graphs allow for the direct visual representation of different variable dependencies, eliminating the need to interpret complex joint probability distributions. Instead joint probability distributions can be rewritten as the product of individual probabilities, greatly simplifying required calculations. The general form of this equation is shown in Equation 2. From a DAG perspective, Equation 2 shows that only probabilities of the parents of vertices need to be calculated to compute the resultant probability of output selection.

$$P(v_1 \dots \dots, v_i) = \prod_{1}^{i} P\big(v_i | parents(v_i)\big) \tag{2}$$

Once a network structure is set, prior probabilities for categories or events within a node must be computed. The prior calculation is often a simple probability calculation. Theoretically, using this formulation, the prior probabilities for categories of a variable will sum to one. In addition to priors, the likelihood of an event given some evidence must also be computed. This work used the Laplace Smoothing likelihood method (MacKay, 1998; Williams, 1995). This method is popular since it considers the probability of seeing a combination of evidence even if an event is not observed in the training data. This is helpful during the testing stage where a network may encounter novel data combinations. The network is considered trained after likelihood calculations are complete. As a result, new points can be passed in for classification. The accuracy of a trained network is gauged by seeing how many testing set data points the network classifies correctly.

Kriging

Kriging models, the first data generation method used for this work, are inherently a way to fit a weighted regression model to a collection of data points (Bohling, 2005; Lovison, 2007). This model can then be used to approximate behavior of the dataset where little to no data is present. The goal of a Kriging model process is to find some function that approximates the behavior of the dataset while minimizing the discrepancy between predicted and expected values.

$$Y(\theta)^* = \sum_{i=1}^{n} \lambda_i * Y(\theta_i) \tag{3}$$

The basic formulation of a Kriging model is shown in Equation 3. $Y(\theta)^*$ represents the expected value of a data point inserted into the model. This prediction is generated using a weighted summation of all the points describing a dataset's behavior. The weights, or λ_i, represent the influence a point in the data set has on a point that is being predicted. Usually these weights decrease the further away a point is from the predicted position θ.

$$\sigma^2(\theta) = E[|Y(\theta)^* - Y(\theta)|^2] \to 0 \tag{4}$$

Ultimately, the goal is to solve for $\lambda_i's$ in Equation 3 that minimize the variance between the predicted and actual values. This difference between expected and actual values, or σ^2, describes how well a model fits the data. The lower the σ^2 value shown in Equation 4, the better the model fit.

Radial Basis Functions

Mathematically similar to Kriging models, Radial Basis Functions (RBFs) are also inherently a way to fit a weighted regression model to a collection of data points (Buhmann, 2000). Like Kriging, RBFs are often used to approximate a dataset's behavior in areas where little to no collected data is present. The basic formulation of an RBF is shown below in Equation 5. The goal of the process is to find weights, or λ_i's, that minimize the difference between the model and the actual points n. This difference is predicted by the basis function φ.

$$Y(\theta) = \sum_{i=1}^{n} \lambda_i \varphi(|\theta - \theta_i|) \quad (5)$$

One commonly used basis function is Gaussian. This type of basis function deals well with noisy data and does a good job of smoothing out noise in a collected data set. The Gaussian formulation is shown below in Equation 6, where ε is a user specified shape factor.

$$\varphi(\theta - \theta_i) = e^{-(\varepsilon|\theta-\theta_i|)^2} \quad (6)$$

Particle Swarm Optimization (PSO)

Simple particle swarm optimization was developed by James Kennedy and Russell Eberhart in 1995 (Eberhart & Kennedy, 1995). The program is modeled after the flocking behavior of birds when in search of food or shelter as described by zoologist Frank Heppner. Flocks of birds use their own experiences and the knowledge of the flock when searching for shelter or food. This behavior forms the basis of the swarm characteristics in PSO. The particles use their own information (pBest) and the flock's knowledge (gBest) to find the optimum solution to an optimization problem. The velocity update function is what influences the behavior of the particles within the swarm, pulling the swarm towards the optimum value. This velocity equation is the heart of the PSO algorithm. Simple PSO was revolutionary when it first debuted but further research suggested that it could be improved upon using weighting factors to alter the original velocity update equation, shown in Equation 7 and 8. These methods such as using a weighting factor, or an inertia weight improve the ability of PSO to find the optimum in a faster and more reliable manner. Constriction PSO improves basic PSO by allowing the swarm to find minimum values more quickly and with greater accuracy (Banks, Vincent, & Anyakoha, 2007; Carlisle & Dozier, 2001). Constriction PSO also eliminates the need to clamp velocity to prevent over exploration of the design space. As such, constriction PSO was the method employed in this paper.

$$\vec{V}_{iter+1,i} = K * \left(\vec{V}_{iter,i} + c_1 * R_p * \left(pBest\vec{X}_i - \vec{X}_i\right) + c_2 * R_g * \left(gBest\vec{X} - \vec{X}_i\right)\right) \quad (7)$$

$$K = {}^{2}\Big/_{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \qquad where: \phi = c_1 + c_2 \; and \; \phi > 4 \quad (8)$$

METHODOLOGY

This section describes how the small data sets were collected and then how Kriging and RBF models were created and used to approximate the size of the data set. Lastly, network training and testing using original and generated data are covered.

Data Collection and Processing

To begin exploring BN for small dataset applications, building a Bayesian Network first required data. Three different datasets were collected. The first dataset was collected from an assembly task using Augmented Reality guided work instructions (Nakanishi, Ozeki, Akasaka, & Okada, 2007; Richardson et al., 2014; Wang, Ong, & Nee, 2016). In the study, participants were asked to assemble a mock aircraft wing made of wood components and metal fasteners. The study setup was designed to mimic a traditional work cell found in an aerospace manufacturing environment. The AR application recorded when a participant moved between steps using a time stamped log files which were parsed to calculate how long participants spent on each step and how long they took to complete the assembly. For detailed information on this study, please see previously published work (Hoover et al., 2016; MacAllister, Gilbert, Holub, Winer, & Davies, 2016; Richardson et al., 2014). The other two datasets, Car Choice and Census, were pulled were pulled from the University of California Irvine (UCI) machine learning database (Asuncion & Newman, 2018). The Car Choice dataset represents a decision model created to describe the decision quality of a buyer's car choice (Bohanec & Zupan, 1997). The Census dataset was pulled from

Table 1. Datasets

Dataset	# of Points	Origin
AR Assembly	75	User Study
Car Choice	1,728	UCI Database
Census	48,842	UCI Database

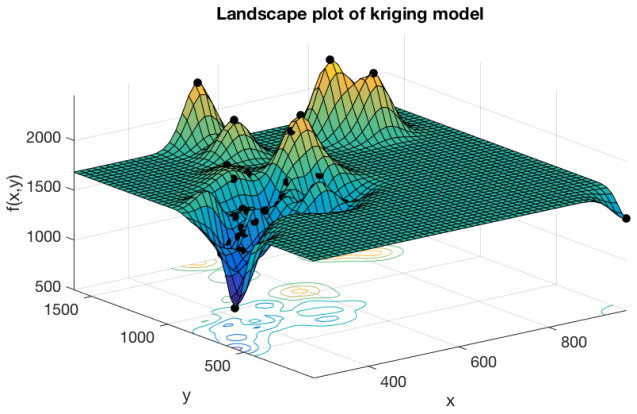

Figure 1. Kriging Model Fit to Limited Training Data

a subset of census responses in the 1990's. The goal of the dataset if to predict the income level of a respondent (Kohavi & Becker, 1996). Table 1 shows the number of points in each dataset. Since the two datasets pulled from UCI were orders of magnitude larger than the AR assembly dataset and since the goal of the work is to study very small datasets, training data was randomly pulled from the Car Choice and Census data. For the Car choice data around 2% was used as training data. For the Census data, a randomly selected .1% subset was used for training. The leftover data was used for testing network accuracy. Since previous work showed that for small datasets the random assignment of testing and training data points can impact accuracy (MacAllister, Winer, & Miller, 2017), each dataset was randomly split into testing and training sets fifteen different times. These fifteen different splits were used to train and test the network to provide an average accuracy measure of performance.

Data Generation

After splitting the data into small training sets and setting aside some data for testing, the next step was to generate additional data. To generate data, the authors decided to utilize meta-models due to their proven use in engineering design. These models allowed the authors to fit a mathematical function to the design space and produce more data as needed. Data generated from the limited training set can then be used to create a BN. Using the testing data to gauge network performance can then show if network accuracy increases when augmenting training with generated data. For is portion of the work, Kriging and RBF models were selected to model the data because of their ability to efficiently describe the behavior of small datasets. This quality has been displayed repeatedly in many optimization publications (Kleijnen, 2009). For each of the fifteen different training datasets both a Kriging and RBF model was fit to the data. Kriging models were fit to each set of datasets using the ooDACE MatLab toolbox (Couckuyt, Dhaene, & Demeester, 2014). Values generated using Radial Basis Functions were created using an RBF MatLab tool box (Chirokov, 2006).

Figure 1 shows an example Kriging model and Figure 2 shows an example RBF model fit to AR Assembly data. The black points in Figure 1 and the red points in Figure 2 are the actual data points. Models fit to these points allow the entire domain to be approximated, especially in areas where little original data exists. These approximations and others like them were used to generate additional training data for a BN. To generate this additional data, the model was randomly sampled 1000 times. These points were then used to train an BN.

Each of the Kriging models used a Gaussian correlation function to build the mathematical representation and to compute the expected vs actual values as shown in Equations 5 and 6. Gaussian correlation functions are popular in metamodeling for engineering applications (Simpson, Peplinski, & Koch, n.d.). For this application, it was used to ensure a smooth fit to the data and minimize the impact any noise the real-world data set contains. In addition,

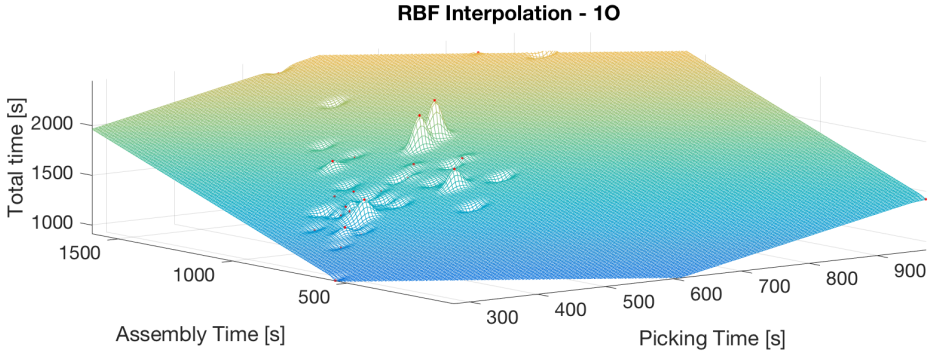

Figure 2. RBF Model Fit to Limited Training Data

ooDACE provides a plot of errors at each point on the model, but these were excluded due to space constraints. The low σ^2 error values in the plots suggest the created models fit the data available relatively accurately. However, looking at the example models and the data points available, there are large areas with limited data. This suggests that in some areas of the model, the approximation may be an extrapolation instead of an interpolation.

Like the Kriging models, each of the RBF models used a Gaussian correlation functions. However, the RBF model was tuned to fit the data using the shape parameter, resulting in a mesh that very closely described the data's behavior. As with the Kriging example model, looking at the RBF model one can see that there are areas with limited data. This suggests that in some areas of the model, the approximation again may be more of an extrapolation than interpolation. In addition, randomly sampling the design space could adversely impact the distribution of data in each category. This could negatively impact the generated data's ability to represent the system by skewing prior probabilities. To deal with this possibility the authors theorized that PSO could help identify the best priors to use for a category. This is explored in greater detail in the results section.

Network Construction

After data were collected, before network training could occur, creating a network DAG structure for each of the datasets was required. The two datasets pulled from the UCI database have been cited in a number of publications specifically looking at creating network structures. For these two datasets, common structures found in academic literature were used (J Cheng, 2001; Jie Cheng, Hatzis, & Page, 2001; Salama & Freitas, 2013). However, since the AR Assembly data was a novel dataset, network construction required preliminarily analysis to understand relationships between variables. Linear regression models were created to show the strength of influence of variables upon each other and used to construct a DAG. An example network structure for predicting time using the AR Assembly data is shown in Figure 3. For more detail on network construction for the AR dataset please see previous work detailing the network construction process (MacAllister et al., 2017). This type of analysis was necessary due to the small data set, meaning research methods that construct network structure could not be used. While research in more automated network construction does not require as much expert input, it relies on having large amounts of data that can be used to learn relationships between variables. However, with small data sets there is often not enough data to make these relationships clear to an automated algorithm. In addition, solely relying on data to construct a network structure is still an active research area and some results indicate that networks constructed only from data are not as accurate as those including expert knowledge (Fenton, 2012; Masegosa & Moral, 2013; Zhou, Fenton, & Neil, 2014). As a result, when using a Bayesian Network approach expert understanding of a process is often necessary to help create a network structure when attempting to use small or even large data sets.

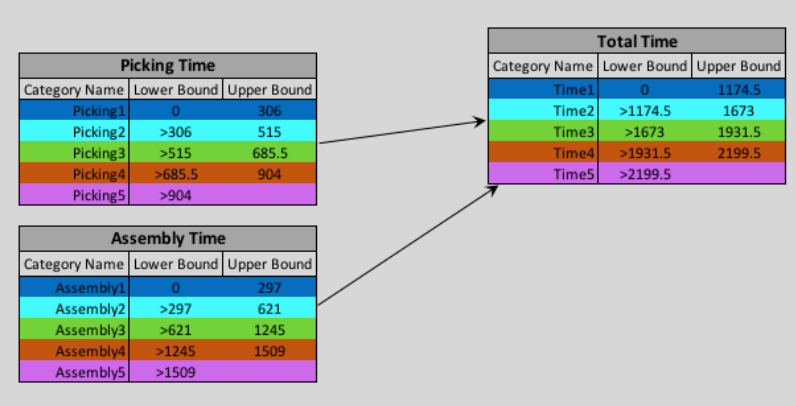

Figure 3. Bayesian Network Structure

For all datasets, Hierarchical Clustering was used to group like values (Kerber, 1992). Grouping the data into clusters or categories makes the likelihood calculations less computationally intensive. In addition, discretized data works better with smaller data sets since there may not be enough data to construct a continuous probabilistic distribution. Example results from the clustering algorithm are shown in network structure in Figure 3. Each vertex has multiple categories, with lower and upper bounds. Discretizing the data requires a participant's recorded numeric value for a variable to be assigned to a category where it fits inside the bounds. Each category houses multiple participant values that fall within its specific assigned range. As a result, instead of continuous values, categorical values are used for training networks.

Training the Bayesian Networks

Training the networks required calculating the likelihood of observing specific evidence states within a predictor category. The predictor vertices varied by network. Using the AR Assembly dataset two networks were constructed. One to predict total completion time and the other to predict the number of errors a participant made. For the Car Choice dataset, the suitability of the choice was the predictor variable. Finally, for the Census dataset income category was the predictor variable. To calculate the likelihood of observing specific evidence, the number of times evidence combinations appeared were calculated for a predictor category. Table 2 is a representative example evidence combination table including frequency for the AR dataset when attempting to predict time. Once an evidence table is constructed for a network training can continue by using the number of observed combinations to calculate likelihood.

Taking a closer look at the evidence counts column in Table 2, it is apparent that the training data is not evenly distributed across the time categories. The vast majority of the observed evidence exists in time category two. This suggests that the network does not have much information on the behavior of data points outside time category two.

Table 2. Likelihood Table for Time Network

Predictor Category	Evidence-Assembly	Evidence-Picking	Evidence Count	Prior
Time1	Assemb2	Picking1	1	0.053
Time1	Assemb3	Picking2	1	
Time2	Assemb3	Picking2	18	0.632
Time2	Assemb3	Picking3	5	
Time2	Assemb1	Picking5	1	
Time3	Assemb3	Picking3	4	0.158
Time3	Assemb3	Picking2	2	
Time4	Assemb3	Picking3	1	0.053
Time4	Assemb4	Picking3	1	
Time5	Assemb3	Picking3	1	0.105
Time5	Assemb4	Picking4	2	
Time5	Assemb5	Picking3	1	

This lack of information outside of time category two and the bulk of the data falling into category two, skewing the priors, suggests that the network will over assign the category that it is most familiar with. The propensity to over assign time category two is a direct result of the limited information available from the small data set. Since there is only a small number of points describing the system's behavior, likelihood values will play only a small role deciding a point's category assignment, allowing the prior to drive assignments. This could end up causing the network to miscategorize points as time category two since it is by far the most frequently represented category and will have the largest prior. The lack of evidence in some categories in Table 2 is characteristic of a small data set. A larger data set would contain more points and a wider variety of points to ensure that all the behavior of the recorded process is captured.

By generating data based on a Kriging or RBF model, a wider variety of evidence states can be represented. Due to space constraints all of those evidence combination tables are not shown for the other three networks. However, for larger data sets, such as those generated using Kriging and RBF, more evidence combinations are generally present. More states are represented in the Bayesian Network could allow it to better predict events it encounters. The next section explores how well Bayesian Networks trained with generated data perform, to see if this theory holds.

RESULTS AND DISCUSSION

After all the steps consisting of splitting data into testing and training sets, and network structure creation the networks were ready for testing. The testing process involved running fifteen different splits of testing and training data for each through one of the four corresponding network structures. Data used for training a network each run was either original data, Kriging generated data, and RBF generated data. The average accuracy of these fifteen runs was used to gauge how accurate each BN was using different types of data. Fifteen runs were used since previous work showed that with small datasets slight changes in the breakdown of testing and training data can impact results (MacAllister et al., 2017). Averaging results across a number of runs minimizes the slight changes due to dataset composition.

Figure 4 shows the testing results when trying to predict total assembly time using original, RBF, and Kriging data. The histogram of accuracy results from the 15 networks trained and tested with random allocations of data also contain box plots describing the distribution of the results. The medians of the box plots in Figure 4 show that the original data trained networks achieve the highest median accuracy, even though RBF and Kriging have two orders of magnitude (1000 vs ~35) more data points. This suggests that even though the Kriging and RBF models were good fits to the data, they don't capture the randomness associated with human operators well. Or it could also be that some of the testing data lay in the extrapolation range of the model or sampling the design space randomly threw off the priors. As a result, a model trained to predict completion time using generated data would not have accurate predictive capabilities.

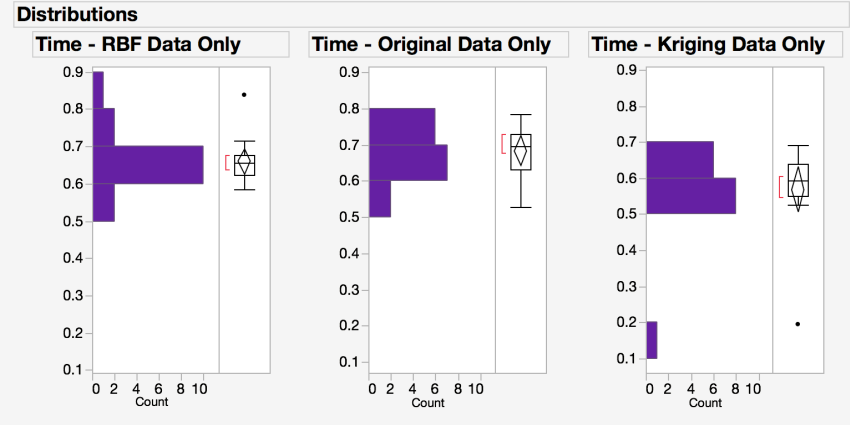

Figure 4. AR Dataset Time Network

Figure 5. AR Dataset Error Network

The second network structure tested for the AR assembly data shown in Figure 5, produces similar results. The box plots show the mean network testing accuracy for original data trained BN is slightly higher than RBF and Kriging. This again suggests that something is missing when generating data.

To see if original data trained networks are more accurate than generated, the results from the second two datasets need to be examined. Histograms and box plots for the car choice dataset shown in Figure 6 indicate that original data produces a network with twice the accuracy of the Kriging and RBF. This suggests that these models, especially the Kriging model, did not generate data that was very good for training a BN, even though the model was an accurate fit to the data. This could point to another factor contributing to decreasing the accuracy of the BN, other than data generation model accuracy. Another peculiar result is the range of network accuracies seen in the RBF metrics. Some datasets result in good training datasets for the BN, but some poor. Digging into the reason behind this and looking for differences could help guide the creation of good datasets rather than poor.

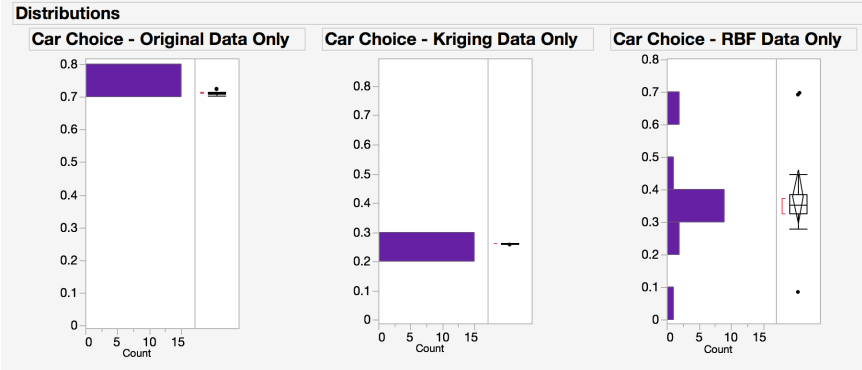

Figure 6. Car Choice Dataset

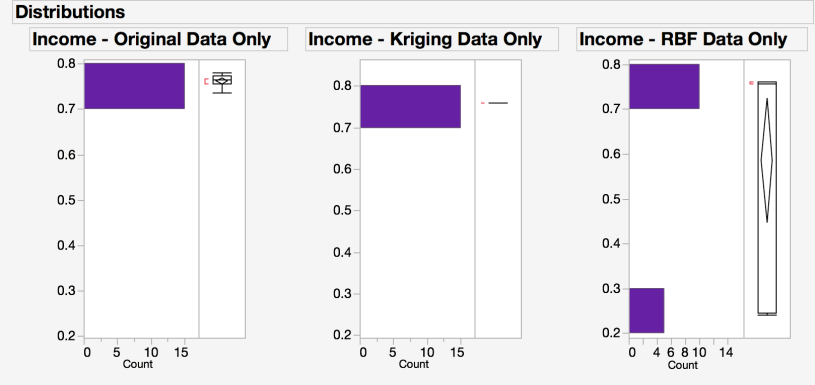

Figure 7. Census Dataset

Moving on to the third dataset in Figure 7, Census data, the mean accuracy for the original and Kriging data are very similar. RBF also generates a large number of accurately trained networks, however, it's average is reduced by a few poorly performing networks.

The results discussed above suggest that generated data on its own is not enough to increase the accuracy of a Bayesian Network trained with small data. When dealing with such small amounts of data, up sampling to such a high degree to increase dataset variety might be damaging the network's ability to produce accurate predictions. The authors hypothesize that this could be due to the fact that when the number of likelihood combinations are increased through randomly sampling the design space, the priors become distorted. This distortion of the priors unbalances Bayes Theorem and reduces the accuracy of the network. The theory is that in order to use the generated data, priors have to be re-computed in a way that can maximize the usefulness of the generated data. In short, the goal is to decouple the priors from the generated data. This will provide more likelihood values using data generation, while still trying to represent the system with priors that maximize the accuracy of the network.

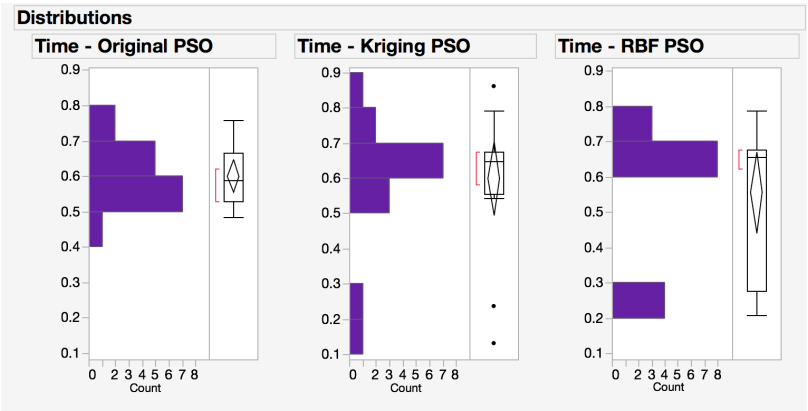

Figure 8. Time Network - PSO Priors

Figure 8 shows the results of using PSO to set prior probabilities for the three different Time Network datasets. The results show that in this case using PSO actually decreases the median accuracy of the networks. However, in some cases the method produces some very accurate networks that surpass the original trained network accuracy. This suggests that sometimes the PSO method could work on a dataset, but not all runs are particularly stable as evidenced by the handful of low-lying results. Even though the method produces some inaccurate networks, the proof of concept showing the presence of high preforming networks is encouraging. These high preforming variants, demonstrating increases over the original data trained networks, could be used as a way to improve classification accuracy. Accuracy gains from the generated data trained networks could potentially help model a process more accurately, saving time and money.

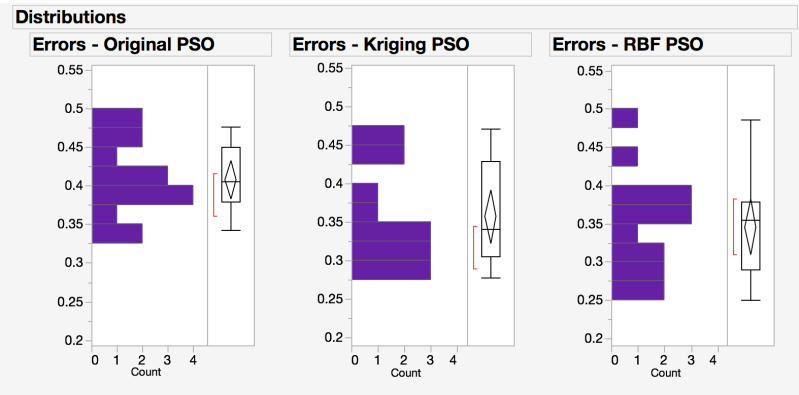

Figure 9. Error PSO Results

Figure 9 shows the results of the error predicting network when the priors are optimized using PSO. Results of PSO prior optimization show accuracy improvements over the non-PSO optimized networks for all three training dataset variations. The median accuracy for original, Kriging, and RBF, are higher than the non-PSO prior optimized networks shown in Figure 5. In addition, maximum classification

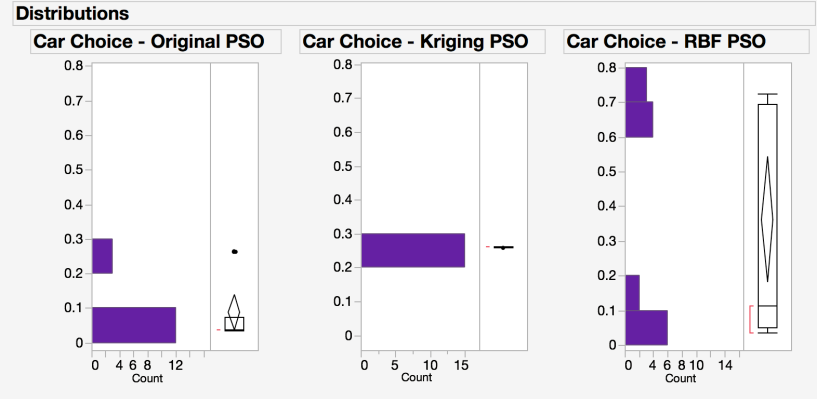

Figure 10. Car PSO Results

accuracy values exceeded previous levels. This suggests that a handful of generated training datasets are promising candidates to use to train the network. Using this experimental data generation approach paired with PSO prior optimization a more accurate network could be created from relatively little data, even when using original data. This could potentially open up more applications for machine learning, specifically those with only small datasets available. Figure 10 shows the PSO results for Car Choice dataset. These results show that using PSO to set priors for the original training data greatly reduces classification accuracy from 71% to around 8%. However, Kriging data trained networks were not impacted by PSO manipulation of priors. Finally, RBF data trained networks saw very slight average accuracy gains and the number of networks with over 60% accuracy increased. It seems with PSO priors RBF generated data is able to surpass the network accuracy set by the original non-PSO data trained network. This again suggests that using PSO coupled with data generation can help improve network accuracy when using small datasets. However, accomplishing these accuracy gains require experimentation and a number of trial runs to produce a good accuracy improving candidate.

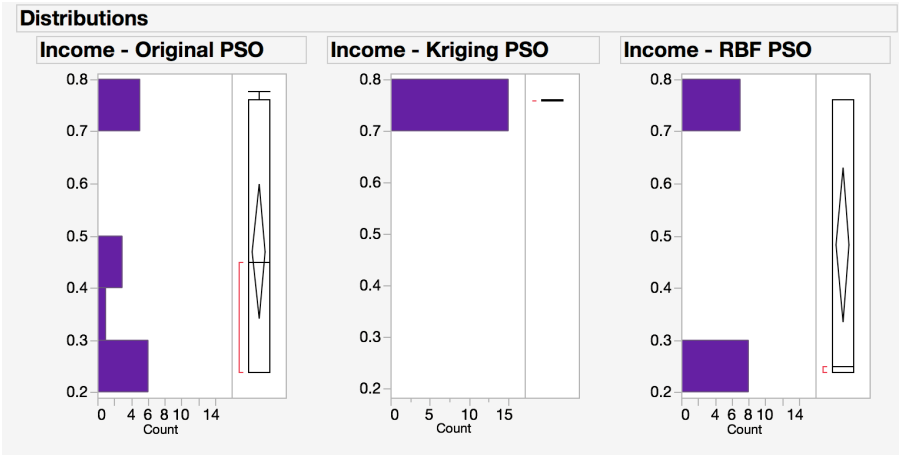

Figure 11. Income PSO Results

Results from the Census dataset predicting a person's income shown in Figure 11 are less clear. These results show that using PSO to manipulate prior accuracy has less of an impact on the average accuracy of the generated data trained networks than the other datasets. This could suggest that the generated network priors are fairly stable and not as sensitive to variation as the original data.

Overall, the results presented above demonstrate a proof of concept showing that in some cases PSO can produce networks that match or exceed classification accuracy of original data trained networks. High accuracy networks tuned using PSO or augmented with generated data could be used as a way to improve prediction models for areas in industry and the military that are currently underserved by traditional ML approaches.

CONCLUSION AND FUTURE WORK

Bayesian Networks are a very powerful tool for modeling and predicting complex relationships between variables. They provide a transparent way to map and understand variable relationships and how changes to networks might impact their accuracy. Their easily understandable network structure paired with flexible Bayesian Statistical methods lends itself well to investigating behaviors associated with small data sets for machine learning. Greater understanding of how to adapt machine learning tools like BN to use with small datasets will ultimately help underserved areas like small volume manufacturing or military applications utilize the powerful predictive analytics of machine learning. The work in this paper took initial steps towards this goal by exploring the feasibility of using Kriging and Radial Basis Function models to generate data for four different Bayesian Networks. The goal of the network was to predict

completion time for workers conducting assembly operations, predict the number of errors an assembly worker made, a buyer's car choice, and the income level of an adult. Data for the project was collected from a human-subjects study that used augmented reality guided work instructions and from the UCI machine learning database. Small amounts of data were used to train the different BNs. Each of these training datasets were each fitted with a Kriging and a Radial Basis Function model. These models were randomly sampled to produce a larger dataset for training. Results showed that generated data, alone, did not increase the accuracy of the trained networks. As a result, Particle Swarm Optimization (PSO) was used to set the priors of the data to gauge if prior optimization could improve network accuracy. Results of these tests showed that using PSO can help increase network accuracy. However, not all network and data combinations see accuracy improvements. Moving forward, the authors will explore the characteristics of the datasets that experience accuracy gains and gauge if generating even greater amounts of data impact network accuracy. The goal of future work is to gain greater understanding of best practices for augmenting small datasets used to train Bayesian Networks. In the end, the work presented above provides the first steps towards coming up with a strategy to begin using Bayesian Networks in real world small dataset problems. However, these initial results should be applied with caution to other applications. The quality of the small dataset and of the models used to generate data can have a large impact on the augmented dataset. If the small dataset is of poor quality generating more data might not improve network performance. When evaluating datasets for potential augmentation it is important practitioners sufficiently test the accuracy of networks trained using generated data to ensure that poorly trained networks are not used to make important decisions.

REFERENCES

Asuncion, A., & Newman, D. (2018). UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml/about.html

Banks, A., Vincent, J., & Anyakoha, C. (2007). A review of particle swarm optimization. Part I: Background and development. *Natural Computing*, *6*(4), 467–484. http://doi.org/10.1007/s11047-007-9049-5

Bayes, T. (1763). An Essay Towards Solving a Problem in the Doctrines of Chances. *Philosophical Transactions*, *53*(1764), 370–418. http://doi.org/10.1093/biomet/45.3-4.293

BBC. (2015). Boeing Delivers Record Number of Aircraft in 2015.

Bernard, J., Chang, T.-W., Popescu, E., & Graf, S. (2015). Using Artificial Neural Networks to Identify Learning Styles. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9112*, 883. http://doi.org/10.1007/978-3-319-19773-9

Biewald, L. (2016). How Real Businesses Are Using Machine learning.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bohanec, M., & Zupan, B. (1997). UCI Car Evaluation Data Set. Retrieved from http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Bohling, G. (2005). Kriging. *Scientist*, (October), 1–20.

Buhmann, M. D. (2000). Radial basis functions. *Acta Numerica*, *9*, 1–38. http://doi.org/10.1017/S0962492900000015

Carlisle, A., & Dozier, G. (2001). An Off-The-Shelf PSO. *Population English Edition*, *1*, 1–6. Retrieved from http://antho.huntingdon.edu/publications/Off-The-Shelf_PSO.pdf

Chen, K. Y., & Wang, C. H. (2007). A hybrid SARIMA and support vector machines in forecasting the production values of the machinery industry in Taiwan. *Expert Systems with Applications*, *32*(1), 254–264. http://doi.org/10.1016/j.eswa.2005.11.027

Cheng, J. (2001). Learning bayesian belief network classifiers: Algorithms and system. *Advances in Artificial Intelligence*. Retrieved from http://www.springerlink.com/index/yqlk1nubp2d5n1kf.pdf%5Cnpapers2://publication/uuid/B128EB3C-A4A1-411D-816F-0E7A8B241E57

Cheng, J., Hatzis, C., & Page, D. (2001). KDD Cup 2001 Report.

Chirokov, A. (2006). Scattered Data Interpolation and Approximation using Radial Base Functions. Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/10056-scattered-data-interpolation-and-approximation-using-radial-base-functions

Couckuyt, I., Dhaene, T., & Demeester, P. (2014). ooDACE Toolbox: A Flexible Object-Oriented Kriging Implementation. *Journal of Machine Learning Research*, *15*, 3183–3186.

De Martino, B., Kumaran, D., Seymour, B., & Dolan, R. J. (2009). Frames, Biases, and Rational Decision-Making in the Human Brain. *Science*, *313*(5787), 684–687. http://doi.org/10.1126/science.1128356.Frames

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39–43. http://doi.org/10.1109/MHS.1995.494215

Fenton, N. (2012). A short story illustrating why pure machine learning (without expert input) may be doomed to fail and totally unnecessary, 1–2.

Hastie, R. (2001). Problems for judgment and decision making. *Annual Review of Psychology*, *52*(1), 653–683. http://doi.org/0066-4308/01/0201-0653

Holzinger, A., Dehmer, M., & Jurisica, I. (2014). Knowledge discovery and interactive data mining in bioinformatics--state-of-the-art, future challenges and research directions. *BMC Bioinformatics*, *15 Suppl 6*(Suppl 6), I1. http://doi.org/10.1186/1471-2105-15-S6-I1

Hoover, M., MacAllister, A., Holub, J., Gilbert, S., Winer, E., & Davies, P. (2016). Assembly Training Using Commodity Physiological Sensors. *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*, (16159), 1–12.

Jacobs, T. L., Garrow, L. A., Lohatepanont, M., Koppelman, S., Coldren, G. M., & Purnomo, H. (2012). *Airline Planning and Schedule Development* (Vol. 169). http://doi.org/10.1007/978-1-4614-1608-1

Kerber, R. (1992). Chimerge: Discretization of numeric attributes. *Proceedings of the Tenth National Conference on Artificial Intelligence*, 123–128. Retrieved from http://dl.acm.org/citation.cfm?id=1867154%5Cnpapers2://publication/uuid/F1C11C06-12F0-4A83-82D3-CC62A52FBE75

Kleijnen, J. P. C. (2009). Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, *192*(3), 707–716. http://doi.org/10.1016/j.ejor.2007.10.013

Kohavi, R., & Becker, B. (1996). UCI Adult Data Set. Retrieved from http://archive.ics.uci.edu/ml/datasets/Adult

Krishnamurthy, T. (2005). Comparison of Response Surface Construction Methods for Derivative Estimation Using Moving Least Squares, Kriging and Radial Basis Functions. *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 1–28. http://doi.org/10.2514/6.2005-1821

Loh, W.-Y. (2014). Fifty Years of Classification and Regression Trees. *International Statistical Review*, *82*(3), 329–348. http://doi.org/10.1111/insr.12016

Lovison, A. (2007). *Kriging*.

Loyer, J.-L., Henriques, E., Fontul, M., & Wiseall, S. (2016). Comparison of Machine Learning methods applied to the estimation of manufacturing cost of jet engine components. *International Journal of Production Economics*, *178*, 109–119. http://doi.org/10.1016/j.ijpe.2016.05.006

MacAllister, A., Gilbert, S., Holub, J., Winer, E., & Davies, P. (2016). Comparison of Navigation Methods in Augmented Reality Guided Assembly. *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*, (16075), 1–14.

MacAllister, A., Winer, E., & Miller, J. (2017). Predicting Manufacturing Aptitude using Augmented Reality Work Instructions Predicting Manufacturing Aptitude using Augmented Reality Work Instructions. In *I/ITSEC 2017* (pp. 1–13).

MacKay, D. J. C. J. C. (1998). Choice of basis for Laplace approximation. *Machine Learning*, *33*(1), 77–86. http://doi.org/http://dx.doi.org/10.1023/A:1007558615313

Malinova, M., & Mendling, J. (2015). BPM - Driving Innovation in a Digital World, 215–227. http://doi.org/10.1007/978-3-319-14430-6

Masegosa, A. R., & Moral, S. (2013). International Journal of Approximate Reasoning An interactive approach for Bayesian network learning using domain / expert knowledge. *International Journal of Approximate Reasoning*, *54*(8), 1168–1181. http://doi.org/10.1016/j.ijar.2013.03.009

Muller, C. (2003). *Reliability Analysis Of the 4.5 Roller Bearing*. Naval Postgraduate School.

Nakanishi, M., Ozeki, M., Akasaka, T., & Okada, Y. (2007). Human factor requirements for applying augmented reality to manuals in actual work situations. In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* (pp. 2650–2655). http://doi.org/10.1109/ICSMC.2007.4413588

Nielsen, T. D., & Jensen, F. V. (2007). *Bayesian networks and decision graphs*. Springer Science & Business Media.

Ong, Y. D., Ato, S. S., Umar, V. K., & Oshino, K. H. (2016). Definition and Verification of Workers ' Aptitude Toward Assembly Tasks in Production Cells, *10*(1), 43–52.

Orloff, J., & Bloom, J. (2014). Comparison of frequentist and Bayesian inference. *MIT OpenCourseware*, 1–7.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Podgorelec, V., Kokol, P., & Rozman, I. (2002). Decision Trees: An overview and Tehir Use in Medicine. *Journal*

of Medical Systems, *26*(5), 445–463. http://doi.org/10.1023/A

Richardson, T., Gilbert, S., Holub, J., Thompson, F., MacAllister, A., Radkowski, R., … Terry, S. (2014). Fusing Self-Reported and Sensor Data from Mixed-Reality Training. In *I/ITSEC* (pp. 1–12).

Rokach, L., & Maimon, O. (2015). *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing Co.

Ropero, R. F., Flores, M. J., Rumí, R., & Aguilera, P. A. (2016). Applications of hybrid dynamic Bayesian networks to water reservoir management, (November), 1–11. http://doi.org/10.1002/env.2432

Rotella, P. (2012). Is Data The New Oil?

Rusu, C., & Rusu, V. (2006). Radial basis functions versus geostatistics in spatial interpolations. *IFIP International Federation for Information Processing*, *217*(1), 119–128. http://doi.org/10.1007/978-0-387-34747-9_13

Salama, K. M., & Freitas, A. A. (2013). Learning Bayesian network classifiers using ant colony optimization. *Swarm Intelligence*, *7*(2–3), 229–254. http://doi.org/10.1007/s11721-013-0087-6

Simpson, T. W., Peplinski, J., & Koch, P. N. (n.d.). METAMODELS FOR COMPUTER-BASED ENGINEERING DESIGN: SURVEY AND RECOMMENDATIONS. *Engineering with Computers*.

Stephenson, T. (2000). An introduction to Bayesian network theory and usage. *Idiap Research Report*, 31. Retrieved from http://ftp.idiap.ch/pub/reports/2000/rr00-03.pdf

Wang, X., Ong, S. K., & Nee, A. Y. C. (2016). A comprehensive survey of augmented reality assembly research. *Advances in Manufacturing*, (July 2015). http://doi.org/10.1007/s40436-015-0131-4

Williams, P. M. (1995). Bayesian Regularization and Pruning Using a Laplace Prior. *Neural Computation*, *7*(1), 117–143. http://doi.org/10.1162/neco.1995.7.1.117

Zhou, Y., Fenton, N., & Neil, M. (2014). Bayesian network approach to multinomial parameter learning using data and expert judgments. *International Journal of Approximate Reasoning*, *55*(5), 1252–1268. http://doi.org/10.1016/j.ijar.2014.02.008