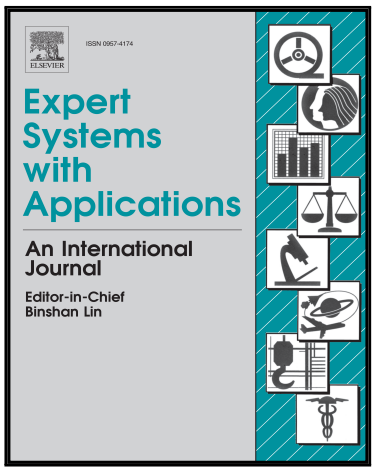

Using High-Fidelity Meta-Models to Improve Performance of Small Dataset Trained Bayesian Networks

Anastacia MacAllister , Adam Kohl , Eliot Winer

PII: S0957-4174(19)30532-9
DOI: https://doi.org/10.1016/j.eswa.2019.112830
Article Number: 112830
Reference: ESWA 112830

To appear in: *Expert Systems With Applications*

Received date: 3 July 2018
Revised date: 7 July 2019
Accepted date: 19 July 2019

Please cite this article as: Anastacia MacAllister , Adam Kohl , Eliot Winer , Using High-Fidelity Meta-Models to Improve Performance of Small Dataset Trained Bayesian Networks, *Expert Systems With Applications* (2019), doi: https://doi.org/10.1016/j.eswa.2019.112830

Highlights:

- Bayesian Networks often cannot be used with small datasets due to accuracy concerns
- Kriging and Radial-Basis Function meta-models are viable options for augmenting datasets
- Bayesian Network accuracy increases when using meta-model generated data

Using High-Fidelity Meta-Models to Improve Performance of Small Dataset Trained Bayesian Networks

Anastacia MacAllister[a], Adam Kohl[a], Eliot Winer[b]

[a]Virtual Reality Applications Center, Iowa State University, IA 50011, USA | Email: amacallister7@gmail.com

[a]Virtual Reality Applications Center, Iowa State University, IA 50011, USA | Email: adamkohl@iastate.edu

[b]Virtual Reality Applications Center, Iowa State University, IA 50011, USA | Email: ewiner@iastate.edu

Abstract

Machine Learning (ML) is increasingly being used by companies like Google, Amazon and Apple to help identify market trends and predict customer behavior. Continuous improvement and maturing of these ML tools will help improve decision making across a number of industries. Unfortunately, before many ML strategies can be utilized the methods often require large amounts of data. For a number of realistic situations, however, only smaller subsets of data are available (i.e. hundreds to thousands of points). This work explores this problem by investigating the feasibility of using meta-models, specifically Kriging and Radial Basis Functions, to generate data for training a BN when only small amounts of original data are available. This paper details the meta-model creation process and the results of using Particle Swarm Optimization (PSO) for tuning parameters for four network structures trained using three relatively small data sets. Additionally, a series of experiments augment these small datasets by generating ten thousand, one-hundred thousand, and a million synthetic data points using the Kriging and RBF meta-models as well as intelligently establishing prior probabilities using PSO.

Results show that augmenting limited existing datasets with meta-model generated data can dramatically affect network accuracy. Overall, the exploratory results presented in this paper demonstrate the feasibility of using meta-model generated data to increase the accuracy of small sample set trained BN. Further developing this method will help underserved areas with access to only small datasets make use of the powerful predictive analytics of ML.

1. Introduction

The rise of commodity sensors and economical computing devices has reduced the cost and effort associated with data collection. A wealth of information can now be collected on manufacturing operations, customers, or even disaster responses. All of this data can provide valuable insights to those willing to analyze it. Organizations and decision makers willing to analyze the relationships between a multitude of variables can gain tremendous insight into previously complicated decisions. These insights can be accomplished by utilizing machine learning techniques. ML methods can help use trends in collected data to make predictions and forecasts, even in novel situations. In addition to being a decision-making aid, these ML tools can help facilitate understanding of systems as a whole, forecasting how different decisions may impact an overall operation. Early adopters of ML techniques like Google, Amazon, and Microsoft are already using machine learning to their competitive advantage by improving their understanding of customers and their products (Biewald, 2016; Reese, 2016; Wilder, 2016). One example of a widely utilized ML approach is Bayesian Networks (BN) (Bayes, 1763). Bayesian Networks combine flexible Bayesian statistical methods with an easy to understand network structure that represents relationships between variables in a concise and transparent

manner (Weber, Medina-Oliva, Simon, & Iung, 2012). The transparency of BNs, due to their easily understood mathematics and compact representation of variable relationships, allows users to understand how changes within a system have the potential to impact the network's behavior. This type of discovery is often limited for other ML methods due to lack of transparency, reducing the ability to improve the methods' accuracy. While ML tools such as Bayesian networks are very powerful, large quantities of data are often required (i.e. hundreds-of-thousands or millions points) to accurately capture behaviors of complex processes.

Even while sensor data is becoming more economical and prevalent, in many common applications sufficient data cannot be collected to utilize ML techniques. In many domains like engineering design, high-precision and/or custom manufacturing or even military exercises, data collection events occur infrequently throughout the year. Consequently, collecting the thousands or millions of data points required for ML is not feasible. Take, for example, aircraft manufacturing. Producing such a complex piece of engineered machinery often involves a number of different collaborators including unionized labor and suppliers. The intertwined performance of the collaborators ultimately impacts the final product. Selecting the right workers for each of the intertwined jobs is important to ensure the success of assembly and manufacturing outcomes. A competitive edge could be gained by ensuring workers with the most suitable skills are assigned to a job (Ong, Ato, Umar, & Oshino, 2016). From a data collection perspective, however, the limited quantity of planes (i.e. 20-30) produced a month does not provide adequate amounts of worker data required for a model to quantify suitability (BBC, 2015). Due to data requirements, for such a low volume task, BNs would not be an ideal option. This is due to the limited amounts of data available to model the network's behavior.

Limited data restricts the model's ability to understand the connections between variables and anticipate how changes might impact the overall system. As a result, low volume processes, that could benefit from the powerful analytics of BN, cannot use the tool because of limited data. However, leveraging meta-models to generate synthetic data from small datasets can benefit these low volume processes by increasing the amount of training data needed for BN.

The presented work takes an exploratory look at how small datasets can be utilized to build accurate BNs. Results provided are intended to be exploratory. As a such, results are not expected to be considered generalizable, rather the goal is to start addressing the limited dataset problem with the hope of leading to more generalizable methods. Hence, the overarching contribution of this work is the investigation of data generation methods to increase predictive performance capabilities of expert systems when little original data exists. The comprehensive aim of this paper contains two parts: (1) evaluate the feasibility of utilizing Kriging and RBF meta-models to increase network performance when subjected to small amounts of training data and 2) investigate the effectiveness of using PSO to intelligently set prior probabilities to avoid the potential skewing of distribution of priors seen when generating synthetic training data.

In the work presented, three different datasets were used. Two of the datasets were collected from a widely used university machine learning database. The other dataset was gathered from a user study looking at the benefits of augmented reality (AR) work instructions. Using these gathered datasets, in total four BNs were created. For the data gathered from the AR study, the goal was to correctly categorize a participant's errors and completion time on the assembly. For the university database collected information, one of the network's goals was to

predict the quality of a buyer's car choice. The second university dataset aimed to classify the income level of a census respondent. A small initial training size of around 40 points made the work challenging. Due to such a small sample size, usually incompatible with BNs, the authors explored the feasibility of using Kriging and Radial Basis Functions (RBF) to augment or increase the original dataset. To gauge the feasibility of this approach, results are presented for networks trained using various amounts of generated data. The sections below give an overview of BNs, describe the data collection methods, describe BN construction, and present network testing results.

2. Background

Analyzing data using statistical methods has for many years helped researchers and practitioners understand the relationships between variables within a dataset. Applications of statistical analysis can be found in areas spanning from scheduling flights to predicting the reliability of a system (Jacobs et al., 2012; Muller, 2003). With the vast amounts of information being collected today, these types of tools become invaluable when attempting to forecast outcomes for decision making using legacy data. Forecasts, built using collected data, have the capability to take into account more interactions between factors than the human mind can comprehend. In addition, it can also provide predictions that are more unbiased (De Martino, Kumaran, Seymour, & Dolan, 2009; Hastie, 2001).

Learning from small datasets can be problematic in terms of predicting specific correlations between input samples and outputs as well as making the model susceptible to overfitting. However, increasing the size of the training set can improve the generalization and stability of

the models (D. Li & Liu, 2009). Generating artificial samples for training has been an effective method to improving learning performance, and has been proven to be mathematically equivalent to incorporating prior knowledge (Niyogi, Girosi, & Poggio, 1998). Oniśko et al. applied a Bayesian network for the diagnosis of liver disorders with a small dataset concerning using using Noisy-OR gates to learn the necessary parameters (Oniśko, Druzdzel, & Wasyluk, 2001). Even though the learned parameters increased network performance, a lack of training samples can yield a less robust model. Yang et. Al proposed a novel virtual sample generation (VSG) method based on Gaussian distribution, which demonstrated the generalization ability of the classifiers on the new training sets can be better than on the original training sets (Yang, Yu, Xie, & Zhang, 2011). However, only a limited number of virtual samples were generated for training, which can hinder learning performance when using small datasets. To avoid the normal distribution assumption, Chen et al. proposed a novel PSO based VSG (PSOVSG) approach to take into consideration the integrated effects of attributes, which resulted in improved accuracy for the forecast model (Chen, Zhu, He, & Yu, 2017). Nevertheless, the PSOVSG method was vulnerable to generating bad samples that lead to a negative impact on model accuracy.

For this 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). Mathematically, Kriging models 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 the behavior of a dataset where little to no data is present. For this application, and in many others, a

Gaussian correlation function was used to ensure a smooth fit to the data and minimize the impact of any noise commonly seen in real-world datasets. RBF, as opposed to Kriging tends to perfectly capture, rather than approximate, dataset behavior. RBFs have great flexibility when matching very random fluctuations in dataset behavior. As a result, RBFs can be tuned to high degrees of accuracy. However, additional data usually necessitates a completely new model because of the high degree of customization.

2.1. Bayesian Statistics

The resurgence of Bayesian statistical models is partially related to the explosion of interest in machine learning research (Pearl, 1988). The theorem underpinning Bayesian statistical methods is provided in Equation 1. Bayes Theorem, as it is known, differs from more previously popular statistics because the prior probability allows background knowledge to be inserted into the probabilistic prediction model (Bayes, 1763). In Equation 1, the portion on the left side is called the posterior probability, or $P(A|B)$. This term represents the probability of some event A occurring given some evidence B is observed. On the left side of the equation is the likelihood and prior probability. The likelihood, $P(B|A)$, represents some evidence state B occurring given some event A. The prior probability, $P(A)$, is the probability of some event A occurring. The last term, $P(B)$, represents the probability of some event B happening at any point in the collected data.

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

Traditionally, a prior term is not included in more commonly used statistical methods. More traditional statistical tools compute probabilities based solely on the collected data (Orloff & Bloom, 2014). However, by adding this prior probability term corrections can be made to observed data allowing it to more accurately model a system. Frequently, the prior probability term is considered expert set parameter. By fusing the data driven likelihood and the expert specified prior, a more accurate probability estimate of some event occurring can be predicted. This combination of terms, expert experience and data driven, results in a robust method for predicting events where only limited information is available (Pearl, 1988).

2.2. Bayesian Networks

When only dealing with a handful of events and variables, Bayes' Theorem can be easily understood. However, as more variables and events are encountered keeping track of all the necessary calculations plus the relationships becomes more difficult. One way to help mitigate this problem is to use Directed Acyclic Graphs (DAGs) to visualize all the variable dependencies (Nielsen & Jensen, 2007; Stephenson, 2000). DAGs consist of elements, known as nodes, comprised of edges and vertices. Nodes are the actual variables that make up the datasets and edges represent the casual relationships between them. Utilizing DAGs, it is no longer necessary to interpret the complex joint probability distributions between variables, rather only the vital relations are represented. This also simplifies the required posterior probability calculations, by only using combinations of individual probabilities. Equation 2 shows the formulation using only the probabilities of individual parent vertices to compute the overall probability value.

$$P(v_1 \dots \dots, v_i) = \prod_{1}^{i} P\left(v_i \middle| parents(v_i)\right) \#(2)$$

Prior probability calculation on the other hand is often more straight forward. Often the method of prior calculation is a simple probability that sums to one across a category of evidence. The last term, likelihood, was computed for this work using the Laplace Smoothing method (MacKay, 1998; Williams, 1995). The Laplace method is popular because it accounts for the possibility of an event even if it is not found in the training data, a benefit when working with small datasets. A network is trained once the structure has been set and the likelihoods and priors have been computed. After training, novel points may be passed into the network to test its accuracy, of which the overall accuracy of a network is judged by how many points it classifies correctly.

3. Methodology

The goal of the methodology section is to explain to the reader the collection of the small datasets and how the Kriging and RBF models were used to generated additional training data. After data collection and creation are described, the methodology then moves to a description of network training and testing.

3.1. Data Collection and Processing

Exploring generated data's impact on BN accuracy, first required data collection. For the work three datasets were collected and analyzed. The AR assembly dataset was collected from a user study where participants were asked to assemble a mockup of an aircraft wing using Augmented Reality (Nakanishi, Ozeki, Akasaka, & Okada, 2007; Richardson et al., 2014; X. Wang, Ong, & Nee, 2016). The mock wing was made of metal fasteners and

wooden components. It was designed to resemble an actual aerospace work cell. During the study the Augmented Reality application collected data like operation duration, number of participant steps, and total completion time. A more detailed description of the study can be found in previous academic publications (Hoover et al., 2016; MacAllister, Gilbert, Holub, Winer, & Davies, 2016; Richardson et al., 2014). The Car Choice and Census datasets were gathered from the machine learning database website at the University of California Irvine (UCI) (Asuncion & Newman, 2018). The Car Choice dataset models a buyer's car choice decision based on quality (Bohanec & Zupan, 1997). The Census dataset was pulled from a subset of census responses in the 1990's. The goal of the dataset is to predict the income level of a respondent (Kohavi & Becker, 1996). The size of each of the datasets is displayed in Table 1.

Table 1. Datasets

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

Notice that the datasets gathered from UCI were much larger than the AR study dataset. Since the focus of the work is working with very small datasets, Car Choice and Census datasets were down sampled. Approximately two percent of the Car Choice data was randomly selected for training. For the Census data, a randomly selected 0.1% subset was used for training. The remaining data was used to test network accuracy. Each of the networks were trained fifteen different times using randomly allocated training datasets. Data was split into testing and training sets fifteen different times because results from previous work showed when using small datasets the assignment of points can impact

network results (MacAllister, Winer, & Miller, 2017). Creating fifteen different networks allowed conclusions to be based off of average performance metrics.

3.2. Data Generation

After splitting the data into training and testing sets, metamodels were created using Kriging and RBF formulations due to their ability to capture non-linear behavior present in datasets. Non-linear behavior is inherent in human performance data, like those used in this research. These types of models allowed the authors to fit a mathematical function to the limited data available. This mathematical description of the datasets behavior was then used to produce more data for training. After the networks were trained with generated data, the testing data was then used to gauge network performance which shows if network accuracy increases when augmenting the training process with generated data. For this 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 Kriging and RBF models were fit to the data.

3.2.1. Kriging

Kriging models were fit to each set of datasets using the ooDACE MatLab toolbox (Couckuyt, Dhaene, & Demeester, 2014). Figure 1 shows a Kriging model fit to the AR assembly data. The black points in Figure 1 are the actual data points, in which the x-axis denotes the picking time(s) and y-axis the assembly time(s). Models fit to these points allow the entire domain to be approximated, especially in areas where little original data

exists. However, in some portions of the model there exists large flat areas where little to no actual data exists.

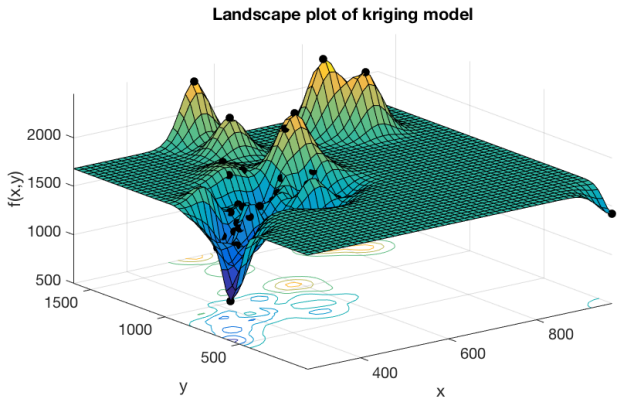

Figure 1. Kriging Model Fit to Limited Training Data

This suggests the approximation in some areas may be more of an extrapolation than interpolation. Consequently, this could mean the model does not adequately capture dataset behavior in this area, which has the potential to negatively impact network accuracy when trained using generated data.

Mathematically, Kriging models 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 a dataset where little to no data is present. The goal of utilizing the Kriging model process is to find some function that approximates the behavior of the dataset while minimizing the discrepancy between predicted and expected values.

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

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 values, 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. $Y(\theta_i)$ represents the function selected to approximate the data's behavior. Common functions include linear, exponential, and Gaussian. Selecting this function is a critically important step in the Kriging process, especially when the data exhibits non-linear behavior. Therefore, it's important to understand the dataset's behavior when determining the approximation function. For example, if non-linear behavior is approximated with a linear function, model predictions could be inaccurate.

$$\sigma^2(\theta) = E[|Y(\theta)^* - Y(\theta)|^2] \rightarrow 0 \ \#(4)$$

Once a function is selected, 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.

Each of the developed Kriging models used a Gaussian correlation function to build the mathematical representation and to compute the expected vs actual values as shown in Equations 3 and 4. Gaussian correlation functions are popular in metamodeling for engineering design applications and surface reconstruction (Krishnamurthy, 2005; Simpson, Peplinski, & Koch, n.d.). For this application, and in many others, it was used to ensure a smooth fit to the data and to minimize the impact of any noise commonly seen in real-world datasets. Gaussian correlation functions also are very adept at fitting any potential non-linearities present in the data. In addition, ooDACE provides a plot of errors

at each point on the model, but these were excluded due to space constraints. The low error values in the plots suggest the created models fit the data available relatively accurately. During testing many of the models produced mean-squared errors of less than one to the negative tenth. However, looking at the example models and the data points available, there are large areas comprised of limited data points. This suggests that in some areas of the model, the mathematical approximation may be inaccurate. This inaccuracy could result in poor quality generated data, negatively impacting network classification when using generated training data.

3.2.2. Radial Basis Functions (RBF)

Values generated using Radial Basis Functions were created using an RBF MatLab tool box (Chirokov, 2006). Figure 2 shows an example RBF model fit to AR Assembly data. The red points in Figure 2 are the actual data points. Like Kriging, Radial Basis Functions (RBFs) are also inherently a way to fit a weighted regression model to a collection of data points (Buhmann, 2000).

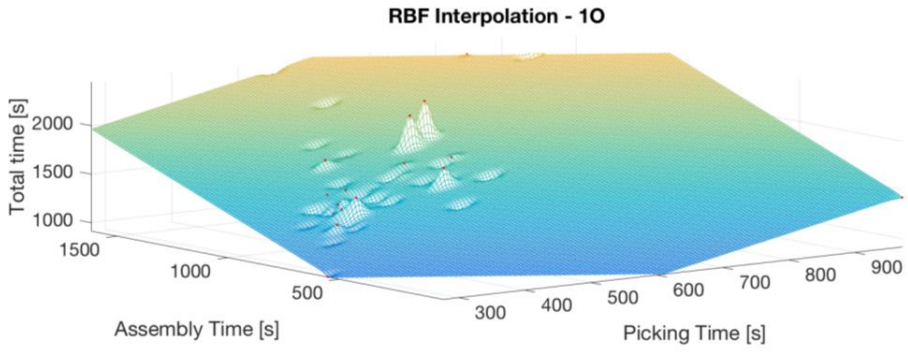

Figure 2. RBF Model Fit to Limited Training Data

However, RBFs incorporate a shape factor that allows the mathematical behavior of the model to be closely tuned to dataset behavior. The basic formulation of an RBF is shown 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, since it deals well with noisy data and does a good job of smoothing out noise in a collected data set. In addition, the Gaussian function is popular due to its ability to mathematically capture non-linearities in a dataset. What differentiates RBF from Kriging is that RBF, in general, tend to perfectly capture rather than approximate dataset behavior. RBFs have great flexibility when matching very random fluctuations in dataset behavior. As a result, RBFs can be tuned to high degrees of accuracy. However, additional data usually necessitates a completely new model because of the high degree of customization. The Gaussian formulation is shown below in Equation 6, where ε is a user-specified shape factor. The shape factor is a contributing factor to the high degree of accuracy exhibited by RBFs. Increasing or decreasing the shape factor changes the width of the selected distribution contributing to how well the formulation fits the available data.

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

Like the Kriging models, each of the RBF models used a Gaussian correlation function. 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 graph of the Kriging model above, the RBF model contains areas with limited real data points. 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.

3.3. Network Construction

After data collection and generation, the next step in the process involved creating a DAG for the four networks. For the two UCI datasets, network structures were pulled from previous academic publications that dealt with network creation (J Cheng, 2001; Jie Cheng, Hatzis, & Page, 2001; Salama & Freitas, 2013). For the AR assembly dataset, there were no preexisting network structures. In order to construct a network preliminary regression analysis was conducted to determine the strongest relationships between variables. The resulting network structure for the Time AR Assembly data network is displayed in Figure 3. The prior probabilities for the network structure, displayed in Figure 3, will be discussed in the following section. See previous publications for greater detail on the network construction (MacAllister et al., 2017).

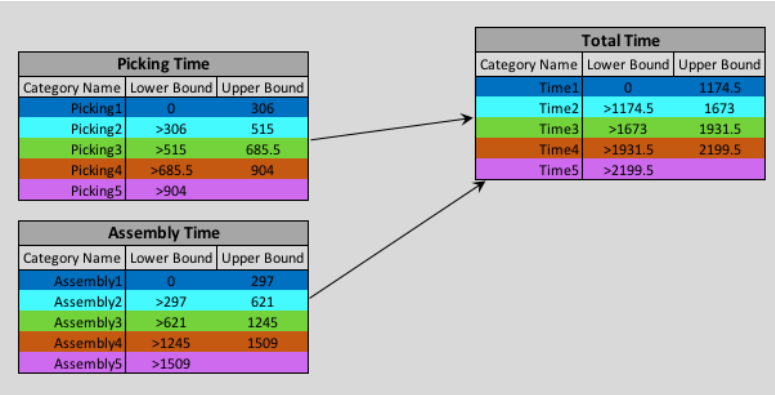

Figure 3. Bayesian Network Structure

To create the categories, like those seen in Figure 3, Hierarchical Clustering was used to discretize the data (Kerber, 1992). Using clustering to group the data with like values, establishing lower and upper bounds for each category, makes the likelihood computation step less resource intensive. Discretized data, also, is more suited to working with small datasets because enough data may not be present to create a continuous distribution model. This is particularly relevant if the domain requires a continuous model. Clustering results in Figure 3 show that each node contains five categories comprised of lower and upper bounds. By discretizing the data, a participant is placed within the category where its specific variable value fits inside the bounds. Each of the categories are assigned to the participants that fall inside its bounds. This discretization means that categories, not continuous variables, are used to train a network.

3.4. Training the Bayesian Networks

Once the data was discretized, training the network necessitated computing the number of observed evidence combinations within a predictor node's categories. Predictor nodes varied by network. The predictor nodes for the two AR user study data networks were errors made on the assembly and completion time. The predictor node for the Car Choice

dataset was the car choice suitability. Finally, for the Census dataset income category was the predictor variable. Table 2 contains the likelihood evidence counts for the AR Time network.

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	

Looking at the counts in Table 2, it's evident that some categories and combinations contain more evidence than others. Specifically, a large portion of the evidence falls into the time two category. As a result, the network knows little about behavior outside of this category. In addition, with the bulk of the data falling into time category two the prior for this category becomes much larger than the others. This could bias the network into assigning time category two due to limited information resulting from a small dataset. Only having a small dataset means that there are very few likelihood evidence combinations. This could result in the prior playing an outsized role in determining classification, resulting in miscategorized points. The issues pointed out in Table 2, like limited evidence states and over represented categories, are very common for small datasets. As the datasets become larger a wider variety of points and evidence states are often introduced. This greater degree of evidence ensures that the BN has enough data of adequately model the system.

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 represented states 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.

3.4.1. Prior Manipulation Using Particle Swarm Optimization (PSO)

Previous work on BN for small datasets suggested that prior probabilities can have a large impact on BN when trained with limited amounts of data (MacAllister, Miller, & Winer, 2018). As a result, an intelligent way to set priors to increase network accuracy was required. The bounds of the problem, like a concrete objective to achieve greater classification accuracy, suggested that optimization methods were well suited to the problem. However, due to the wide variety of methods available, care had to be taken to select the right one (Arbelaez Garces, Rakotondranaivo, & Bonjour, 2016; Jin & Rahmat-Samii, 2007; Konak, Coit, & Smith, 2006; Marler & Arora, 2004; Martins & Lambe, 2013; Padovan & Manzan, 2014; Thornton, Hutter, Hoos, & Leyton-Brown, 2012; G. G. Wang & Shan, 2007). Ultimately, Particle Swarm Optimization (PSO) was the method selected after reviewing the options. PSO was selected since its characteristics aligned well with the unknown design space, lack of problem constraints, and the singular goal of increasing classification accuracy.

PSO was initially conceived in 1995 by James Kennedy and Russell Eberhart (Eberhart & Kennedy, 1995). At a high level, the PSO method mimics the flocking behavior of birds. When in search of food or shelter, birds use both their own experience and the experience of the flock to fulfill their goals. The combination of personal and group knowledge to accomplish a goal is the inspiration of the PSO routines swarm behavior. Particles in the swarm use both the flock knowledge, called the global best (gBest), and their own knowledge (pBest) to find the best solution to a problem. As the knowledge of gBest and pBest changes the particles change their search paths. Overtime this sharing of information guides the swarm to the best-known solution to a problem. The equation that updates the search path, known as the velocity equation, is the cornerstone of PSO. Since its debut in 1995, PSO has been the topic of much research and improvement. More recent research has shown the accuracy of the method can be improved by adding weighting and constriction factors to the velocity equation (Banks, Vincent, & Anyakoha, 2007; Carlisle & Dozier, 2001). The velocity equation with weight and constriction factors used in this paper is shown in Equation 6 and 7. For this work, the overall optimization goal of the swarm was to select priors that resulted in the highest number of correct category assignments within the training dataset.

$$\vec{V}_{iter+1,i} = K * \left(\vec{V}_{iter,i} + c_1 * R_p * \left(pBest\vec{X}_i - \vec{X}_i\right) + c_2 * R_g * \left(gBest\vec{X} - \vec{X}_i\right)\right) \quad \#(6)$$

$$K = {}^{2}\Big/_{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \qquad where: \phi = c_1 + c_2 \; and \; \phi > 4 \qquad \#(7)$$

When implementing the PSO method, the swarm was represented by a multidimensional vector, in which the dimensions correspond to prior probabilities for a predictor node. Take the Time network, a particle for this network would contain a 5D vector. This is because the Time network has five different categories a participant can fall into, thus five prior probabilities. The swarm as a whole is made of up these individual particles, each representing a potential combination of priors. The results presented below use a swarm size of ten particles. Each of these particles are repeatedly tested to gauge its accuracy. This accuracy is then used to update the pBest and gBest values, which drive the swarm's behavior through the update function shown in Equation 6. The update process repeats until the PSO method reaches an accuracy threshold or a set number of updates occur. At the termination of the update loop, the prior values from the best performing particle are selected. In this case, the best performing particle produces the highest network accuracy when using the training dataset. To gauge the overall network accuracy, the optimized priors are used to classify the testing data.

4. Results and Discussion

Following data preparation and creation of the BN topologies, the networks were ready for testing. This involved running fifteen different datasets, each containing training and testing data, for each one of the four corresponding BNs constructed. Data used for training a network during each run was either original, Kriging generated, RBF generated, or a combination of original and generated. The average accuracy of these fifteen runs was used to gauge how precise each BN performed. Average results across a number of runs were used since previous work suggested that when using small datasets the distribution of data has an effect on

classification accuracy (MacAllister et al., 2017). By taking the average classification accuracy the impact of these slight changes can be minimized when assessing overall outcomes.

The first portion of results looks at how different proportions of generated and original data mixed together in the training dataset can impact network accuracy. A 25% (~10 generated pts.), 50% (~20 generated pts.), and a 50/50 (~40 generated pts.) mix of generated data points were added to the approximately 40 original training data points to create the test sets.

The testing results of the first network tested are shown in Figure 4. Each figure contains a histogram of the overall accuracy for the 15 different test runs using various amounts of generated and original data. In addition, the graphs contain a box plot that shows the distribution and the median of the results. For the Time network trained with the AR dataset, using only original testing and training data, the baseline maximum was 78%, the maximum median was 69%, and the standard deviation of the results was 0.07. The baseline result came from training and testing a network using only original data points. Using these baseline accuracy metrics, improvements resulting from adding generated data can be gauged. Results in Figure 4 show that both Kriging and RBF generated data added to original training data could help increase network accuracy slightly. Specifically, Kriging variations 25% and 50% increased maximum accuracy to 80.5%. For RBF, the 50/50 variation increased accuracy to 80.5%, slightly over the baseline of 78%. However, both generated data types increased the standard deviation of results. These results show that some networks produce higher accuracy values, but not all trained networks are more accurate than the baseline. In order to increase classification accuracy, these high accuracy networks could be hand selected and used as a classifier and the lower performing networks discarded. Practically speaking, fifteen networks

are not needed in the real-world. It is only necessary to have one highly accurate network to perform classifications. By adding a mix of generated data to the original data, such a high-quality network can be produced, potentially saving significant amounts of time and money for industry.

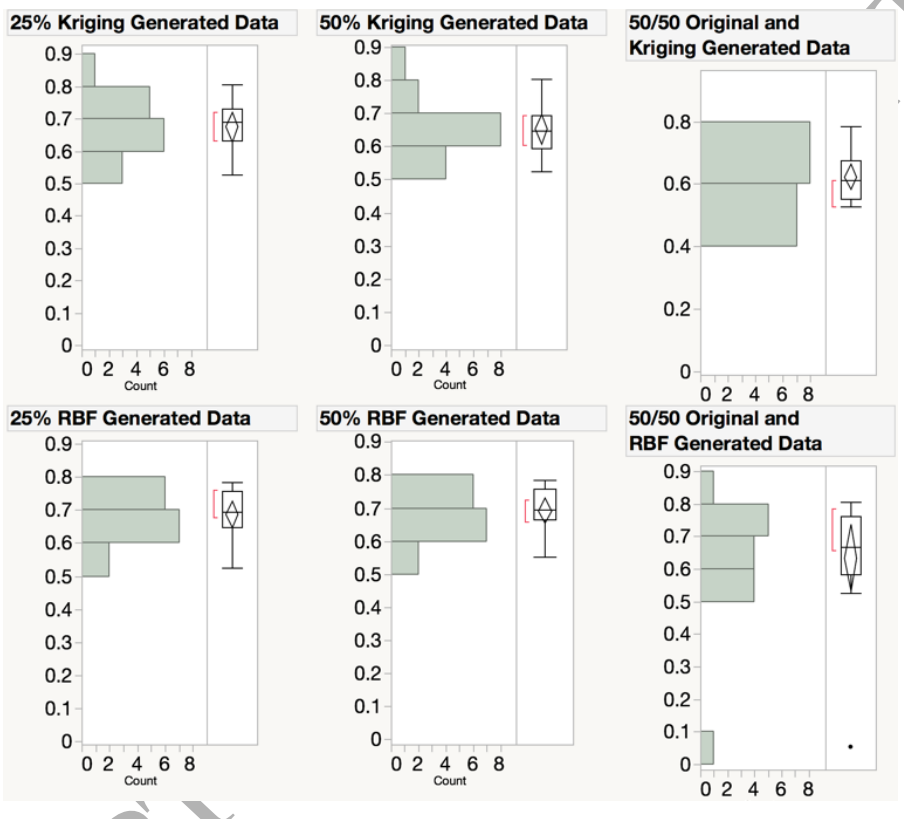

Figure 4. Time Network - Original and Generated Testing Results – Max: 80.5%; Max Median: 69.4%; Max STD: 0.18

The results above show that adding generated data to original training data can help increase accuracy. However, when generating data there is the potential to skew the prior distribution. As such, the authors wanted a way to intelligently set the priors in a way that maximizes the classification accuracy of the networks. The method selected for this task was PSO. The goal of the method is to find prior probabilities that maximize the classification

accuracy of a network. This method was selected because its' characteristics fit well with the problem formulations unbounded optimization requirements.

Figure 5 shows the results of using PSO to attempt to optimize priors for the Time network. Although there are still accuracies above the previous threshold of 78%, PSO does not seem to increase the accuracy of the network over and above the previous results. It actually, decreases the median network accuracy and increases the standard deviation of the results. This suggests that the PSO method in this case is not well suited to help provide additional accuracy gains.

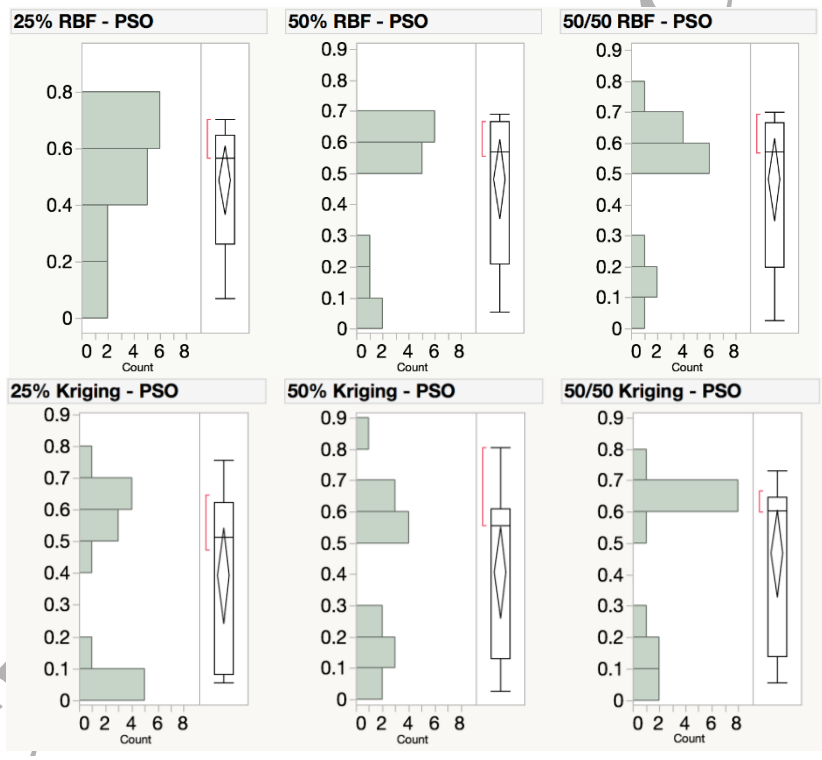

Figure 5. Time PSO Network - Original and Generated Testing Results – Max: 80.5%; Max Median: 60.5%; Max STD: 0.27

Results for the next network, Car Choice, are shown in Figure 6. For this network, academic literature suggests that when using a naïve network structure a reasonable accuracy is around

86% (Jie Cheng & Greiner, 1999; Jie Cheng et al., 2001). The results in Figure 6 show that using only around two percent of the data a network can achieve a maximum accuracy of 73%. This result is encouraging, showing that even when only small amounts of data are available, a somewhat accurate network can be created. Unlike the Time network, however, network accuracy seems to degrade as more generated points are added to the training dataset (following the histograms from left to right). This could suggest that after a certain point, adding additional generated data may stop increasing classification accuracy.

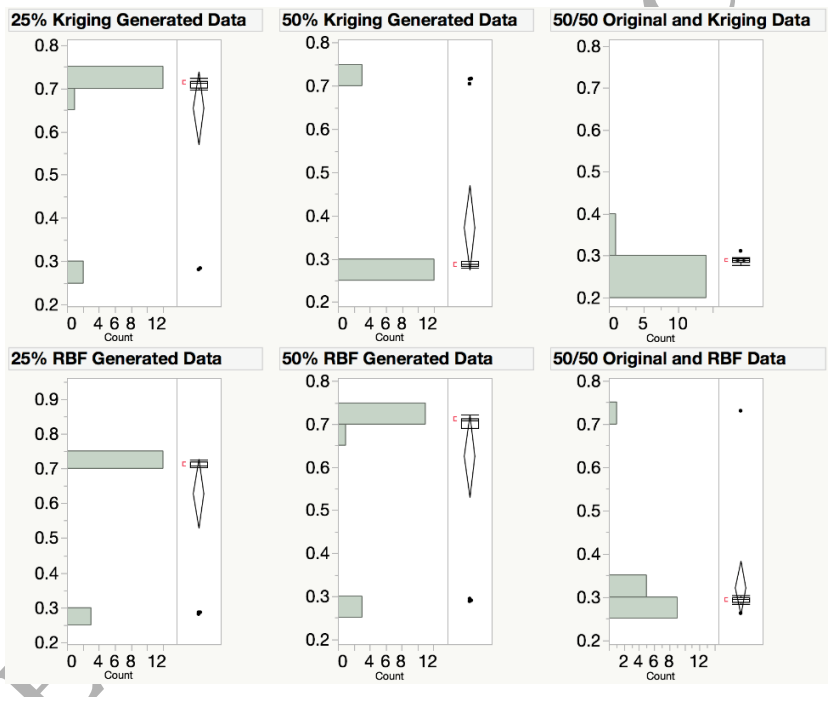

Figure 6. Car Choice Network - Original and Generated Data Testing Results - Max: 73%; Max Median: 71.2%; Max STD: 0.18

Looking at the PSO results for the Car Choice network in Figure 7, it appears that the number of high performing networks increases for the 50/50 variant but does not increase the overall accuracy metrics. This coupled with the lack of accuracy gains using PSO seen in the Time network results could suggest that the PSO formulation needs further modifications.

Currently, the PSO implementation used is fairly basic. Modifying the formulation to include additional terms that give higher weight to better represented categories could help improve results.

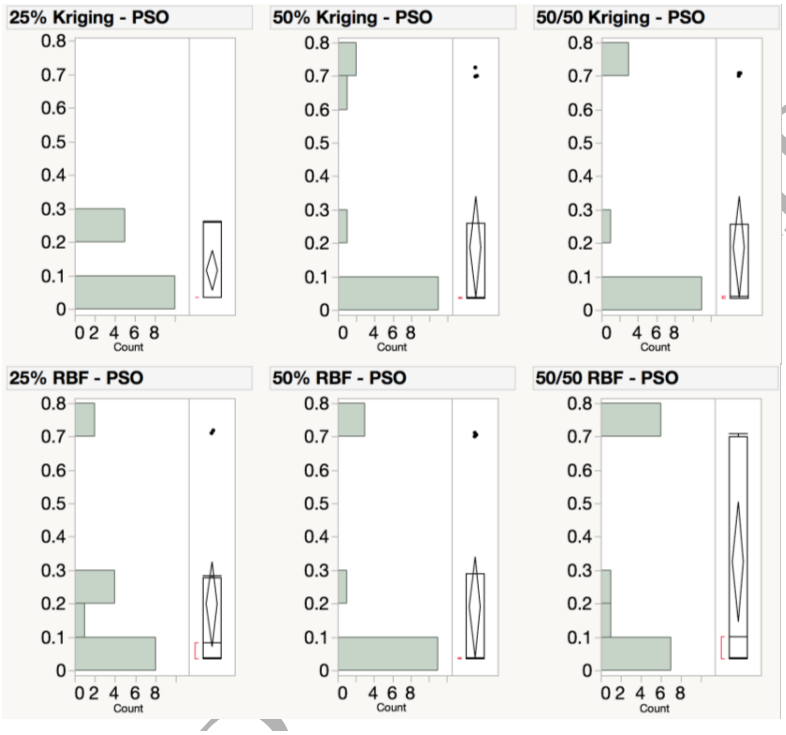

Figure 7. Car Choice Network - Original and Generated Data Testing Results - Max: 72.5%; Max Median: 10.2%; Max STD: 0.32

Due to space constraints the distributions of the other two networks, Errors and Census, are not shown. However, Table 3 shows the maximum accuracy results for all networks along with baseline network accuracies when only using original data for training and testing. The table shows that for Time and Error networks, mixing generated and original data can slightly improve accuracy. For the Time network, adding 25% Kriging generated data to the original approximately 40 training points slightly improved the classifications of the network. Results from testing the Error network show that adding 50% RBF generated data and manipulating the priors using PSO increased network classification accuracy.

Table 3. Maximum Accuracy Results from Generated and Original Training Mix

	Baseline Maximum	Baseline Median Max	Generated Maximum	Generated Median Max
Time	78% (Original)	70% (Original)	80.5% (25% Kriging)	69.4% (25% RBF)
Errors	47.7% (Original PSO)	40.5% (Original PSO)	51.7% (50% RBF PSO)	38.7% (25% RBF PSO)
Car Choice	72.4% (Original)	71.2% (Original)	73% (50/50 RBF)	71.7% (25% Kriging)
Census	78% (Original)	76.4% (Original)	78% (25% Kriging/RBF)	76.2% (Multiple)

Overall, the error testing accuracy is much less than the other three networks. This could suggest that the error network structure does not describe relationships between the data well. While the accuracy results for errors are less than hoped, the network and data still are representative of a real-world problem and worth including in the results. Results for all four network structures, while not overwhelming, do show that it is feasible to use generated data to increase network classification accuracy. In addition, it shows that in some cases the data generation method, used to increase accuracy, might depend on the network structure, since Time and Errors show maximum accuracy gains from two different forms of data generation.

While the results above show promise, the next step is to gauge if generating even greater amounts of data improves accuracy further. Since the approximations (Kriging or RBF) were already available as analytical expressions, creating more data is a real-time operation. For this section of the results 10,000, 100,000, and 1,000,000 data points were generated using Kriging and RBF generated models. Original data was not added to the training set, since the order of magnitude increases in training data means the influence of the original would have been minimal or non-existent.

Figure 8 shows the results of generating varying amounts of synthetic data to train the fifteen different networks. Each of the three RBF training size variants were able to produce a network that was 83.7% accurate. A slight increase over the previous best of 80.5% and the original baseline of 78%. However, the results suggest that for the Time network generating additional data might not prove beneficial.

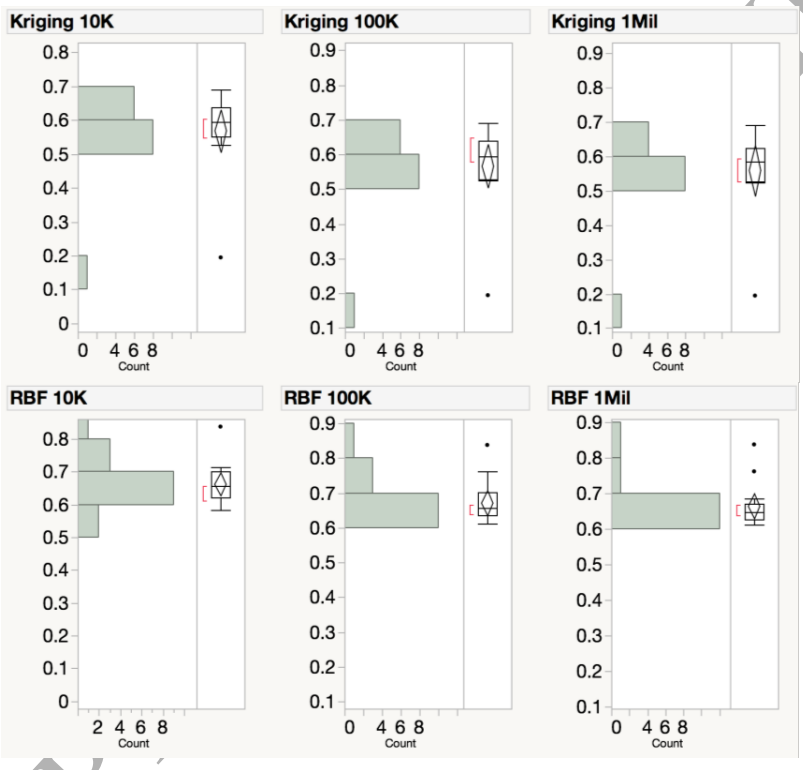

Figure 8. Time Network - 10K, 100K, and 1 Million Generated Points - Max: 83.7%; Max Median: 65.7%; Max STD: 0.12

Figure 9 shows the results of using PSO to attempt to tune the priors. Results show that the tuning was able to produce a maximum accuracy of 82.5%, an improvement over the baseline as well as the previous mixed original generated data network. However, this maximum is less than the 83.5% obtained by the 10K, 100K, and 1 million RBF generated data trained networks.

This again suggests the PSO method, as formulated, is not as beneficial as generating additional data. The increase in standard deviation for all the PSO network results also alludes to the fact that the PSO method may require improvements.

Figure 9. Time PSO Network - 10K, 100K, and 1 Million Generated Points - Max: 82.5%; Max Median: 61.9%; Max STD: 0.27

The results for the next dataset, Car Choice, are shown in Figure 10. For this network, as the number of data points increases so does the median accuracy for both Kriging and RBF, the opposite of the results seen for the Time networks. This could be due to the larger number of causal variables in the Car Choice network (six) than the Time network (two). Greater numbers of variables often require more data to establish the behavior of the network. For the Car Choice network, using one-hundred thousand Kriging generated points a maximum accuracy of 92.4% can be reached, surpassing the 86% accuracy see in previous academic work (Jie Cheng & Greiner, 1999; Jie Cheng et al., 2001). This result is encouraging because it shows that it is

feasible to use meta-models to generate additional training data for BN. Potentially, allowing refinements to this method to open up ML techniques to underserved areas with only small amounts of data.

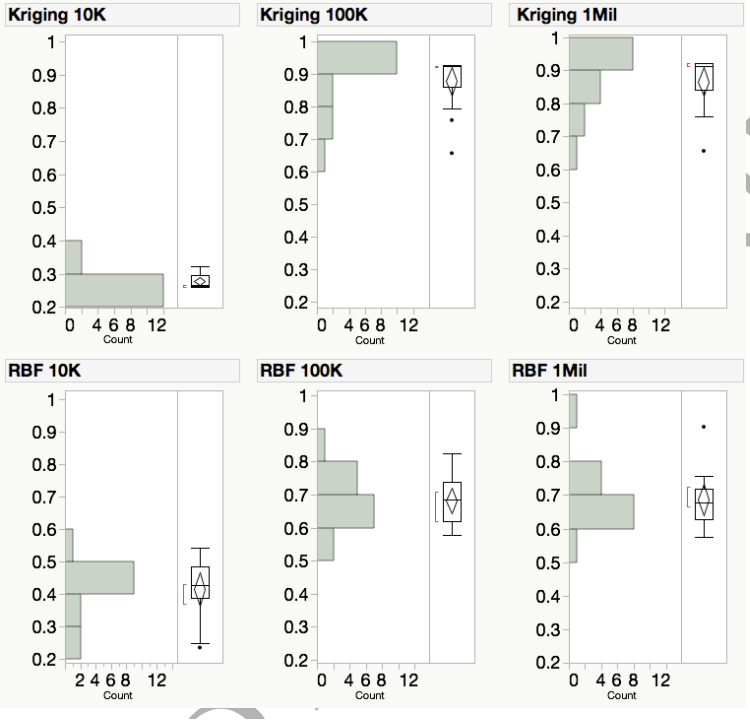

Figure 10. Car Choice Network - 10K, 100K, and 1 Million Generated Points - Max: 92.4%; Max Median: 92.1%; Max STD: 0.08

The PSO results for Car Choice, shown in Figure 11, are similar to those for the Time network. While the maximum accuracy for all runs was near the non-PSO maximum accuracy, using PSO seems to greatly increase the standard deviation of the results, as illustrated by the box plots. Again, this degradation in behavior when PSO is applied suggests that an alternate formulation should be investigated. With the current formulation, all prior categories are given the same weight. If the PSO method was adjusted to weight more common categories more heavily during the tuning of priors, network accuracy might be able to be increased.

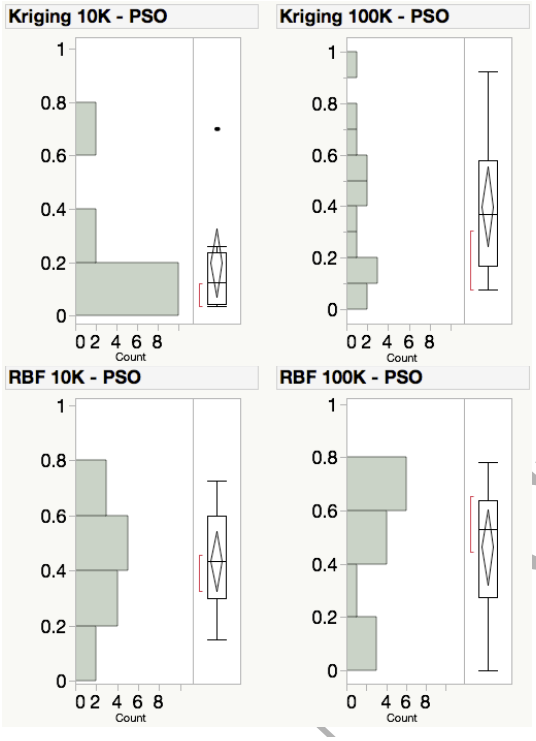

Figure 11. Car Choice PSO Network - 10K, 100K, and 1 Million Generated Points - Max: 92.1%; Max Median: 52.9%; Max STD: 0.27

Again, due to space constraints the graphs for Error and Census networks are not displayed. However, Table 4 shows the maximum accuracy results for all networks tests along with baseline network accuracies when only using original data for training and testing. The table shows that when the mix of generated original data and large-scale data generation are tabulated Time, Error, and Car Choice experience accuracy gains. Specifically, Car Choice sees the highest gain, jumping from around 72% to 92%.

Table 4. Maximum Accuracy Results

	Baseline Maximum	Baseline Median Max	Generated Maximum	Generated Median Max
Time	78% (Original)	70% (Original)	83.7% (10K RBF)	69.4% (25% RBF)
Errors	47.7% (Original PSO)	40.5% (Original PSO)	51.7% (50% RBF PSO)	38.7% (25% RBF PSO)
Car Choice	72.4% (Original)	71.2% (Original)	92.5% (100K Kriging)	92.1% (100K Kriging)
Census	78% (Original)	76.4% (Original)	78% (25% Kriging/RBF)	76.2% (Multiple)

One theory that did not show promise, though, was using PSO to tune prior probabilities. In general, the results show that PSO does not surpass the accuracy metrics of using purely generated data. This type of result when using PSO was also seen in previous work (MacAllister et al., 2018). In addition, PSO tuning priors seem to increase the standard deviation of results and reduce the median accuracy value of the 15 testing runs, except for the Error network. This increase of poorly performing networks coupled with the existence of high performing networks leads the authors to believe that a more sophisticated method of PSO is required to manipulate the priors, one that takes into account more complex relations between variables.

5. Conclusions and Future Work

Bayesian Networks are a very powerful tool for modelling 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 has taken 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 from each of these datasets were used to train the different BNs. Each of these training datasets were fitted with a Kriging and a Radial Basis Function model. Once models were created, they were randomly sampled to produce a larger dataset for training. The four networks were then tested under multiple conditions including the use of PSO to tune network parameters. The first set of results looked at how varying the proportion of generated to original training data would impact network accuracy. Results showed that in some cases generated data could increase the accuracy of the trained networks. In addition, it showed that the varying quantities of original to generated data could also impact the classification accuracy.

From here, the authors generated larger amounts of data. Networks trained using ten thousand, one-hundred thousand, and a million data points were tested. Results showed that depending on the data set, increasing amounts of data did help increase accuracy for more

complex network structures. However, generating too many data points for datasets with a low number of casual variables resulted in decreased performance, which could be a result of overfitting the network. Therefore, it may not always be advantageous to generate additional synthetic training data when trying to increase performance.

Results from tuning network parameters using PSO showed that it can help to produce accurate networks that improve on baseline original data only performance. This is in line with previous research presented by Lessmann et al (Lessmann, Caserta, & Montalvo, 2011). However, Lessmann et al only used a single dataset for evaluation as multiple datasets and trial runs used in this paper. Another interesting observation was when using PSO, the median prediction of the BN lowered while the standard deviation increased. This result leads the authors to believe that an alternate PSO formulation taking into account more information about the parameters is necessary to see further accuracy enhancements. The authors also demonstrated the utility of meta-models to generate synthetic data from small datasets on a real case study compared to other published works in expert systems (D. C. Li, Lin, & Peng, 2014; D. Li & Liu, 2009). Overall, the exploratory results presented in this paper demonstrate the feasibility of using meta-model generated data to increase the accuracy of small sample set trained BN.

There are several areas of future work. First, is exploring a better means of identifying the ideal number of synthetic data points to generate for maximum network performance. A second would be investigating how an improved PSO formulation could more accurately establish prior probabilities. Third, the authors will investigate different ML frameworks that can take advantage of these meta-model data generation methods to increase network

performance when data limitations hinder the use of these expert systems. Lastly, will be a more rigorous investigation to determine the tradeoff of generating synthetic data versus manipulating prior probabilities.

Credit Author Statement

First Author: Anastacia MacAllister
- Conceptualization
- Formal analysis
- Methodology
- Investigation
- Writing – original draft

Co-Author: Adam R Kohl
- Data curation
- Writing – review & editing
- Visualization
- Validation
- Software

Corresponding Author: Eliot Winer
- Funding Acquisition
- Supervision
- Resources
- Project administration

Confl icts of Interest Statement

None

References

Arbelaez Garces, G., Rakotondranaivo, A., & Bonjour, E. (2016). Improving users' product acceptability: an approach based on Bayesian networks and a simulated annealing algorithm. *International Journal of Production Research*, *54*(17), 5151–5168. https://doi.org/10.1080/00207543.2016.1156183

Asuncion, A., & Newman, D. (2018). UCI Machine Learning Repository.

Banks, A., Vincent, J., & Anyakoha, C. (2007). A review of particle swarm optimization. Part I: Background and development. *Natural Computing*, *6*(4), 467–484. https://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. https://doi.org/10.1093/biomet/45.3-4.293

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

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

Bohanec, M., & Zupan, B. (1997). UCI Car Evaluation Data Set.

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

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

Carlisle, A., & Dozier, G. (2001). An Off-The-Shelf PSO. *Population English Edition*, *1*, 1–6.

Chen, Z. S., Zhu, B., He, Y. L., & Yu, L. A. (2017). A PSO based virtual sample generation method for small sample sets: Applications to regression datasets. *Engineering Applications of Artificial Intelligence*, *59*(January), 236–243. https://doi.org/10.1016/j.engappai.2016.12.024

Cheng, J. (2001). Learning bayesian belief network classifiers: Algorithms and system. *Advances in Artificial Intelligence*.

Cheng, J., & Greiner, R. (1999). Comparing Bayesian Network Classifiers. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 101–108.

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

Chirokov, A. (2006). 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. https://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. https://doi.org/10.1109/MHS.1995.494215

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

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). https://doi.org/10.1007/978-1-4614-1608-1

Jin, N., & Rahmat-Samii, Y. (2007). Advances in particle swarm optimization for antenna designs: Real-number, binary, single-objective and multiobjective implementations. *IEEE Transactions on Antennas and Propagation*, *55*(3 I), 556–567. https://doi.org/10.1109/TAP.2007.891552

Kerber, R. (1992). Chimerge: Discretization of numeric attributes. *Proceedings of the Tenth National Conference on Artificial Intelligence*, 123–128.

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

Kohavi, R., & Becker, B. (1996). UCI Adult Data Set.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, *91*(9), 992–1007. https://doi.org/10.1016/j.ress.2005.11.018

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. https://doi.org/10.2514/6.2005-1821

Lessmann, S., Caserta, M., & Montalvo, I. (2011). Tuning metaheuristics : A data mining based approach for particle swarm optimization. *Expert Systems With Applications*, *38*(10), 12826–12838. https://doi.org/10.1016/j.eswa.2011.04.075

Li, D. C., Lin, L. S., & Peng, L. J. (2014). Improving learning accuracy by using synthetic samples for small datasets with non-linear attribute dependency. *Decision Support Systems*, *59*(1), 286–295. https://doi.org/10.1016/j.dss.2013.12.007

Li, D., & Liu, C. (2009). A neural network weight determination model designed uniquely for small data set learning. *Expert Systems With Applications*, *36*(6), 9853–9858. https://doi.org/10.1016/j.eswa.2009.02.004

Lovison, A. (2007). *Kriging*. https://doi.org/10.1038/nn1840

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., Miller, J., & Winer, E. (2018). Manipulating Prior Probabilities to Improve the Performance of Bayesian Networks with Small Datasets. *Submitted to Computers in Industry (Under Review)*, 1–13.

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. https://doi.org/http://dx.doi.org/10.1023/A:1007558615313

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*(6), 369–395. https://doi.org/10.1007/s00158-003-0368-6

Martins, J. R. R. a., & Lambe, A. B. (2013). Multidisciplinary Design Optimization: A Survey of Architectures. *AIAA Journal*, *51*(9), 2049–2075. https://doi.org/10.2514/1.J051895

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). https://doi.org/10.1109/ICSMC.2007.4413588

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

Business Media.

Niyogi, P., Girosi, F., & Poggio, T. (1998). Incorporating Prior Information in Machine Learning by Creating Virtual Examples. *Proceedings of the IEEE*, *86*, 2196–2209.

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.

Oniśko, A., Druzdzel, M. J., & Wasyluk, H. (2001). Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates. *International Journal of Approximate Reasoning*, *27*(2), 165–182. https://doi.org/10.1016/S0888-613X(01)00039-1

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

Padovan, R., & Manzan, M. (2014). Genetic optimization of a PCM enhanced storage tank for Solar Domestic Hot Water Systems. *Solar Energy*, *103*, 563–573. https://doi.org/10.1016/j.solener.2013.12.034

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

Reese, H. (2016). Machine Learning: The Smart Person's Guide.

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).

Salama, K. M., & Freitas, A. A. (2013). Learning Bayesian network classifiers using ant colony optimization. *Swarm Intelligence*, *7*(2–3), 229–254. https://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.

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2012). Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. https://doi.org/10.1145/2487575.2487629

Wang, G. G., & Shan, S. (2007). Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, *129*(4), 370. https://doi.org/10.1115/1.2429697

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

Weber, P., Medina-Oliva, G., Simon, C., & Iung, B. (2012). Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*, *25*(4), 671–682. https://doi.org/10.1016/j.engappai.2010.06.002

Wilder, C. (2016). Rise Of The Machines Part 1: Google And Microsoft Stake Their Claims.

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

Yang, J., Yu, X., Xie, Z. Q., & Zhang, J. P. (2011). A novel virtual sample generation method based on Gaussian distribution. *Knowledge-Based Systems*, *24*(6), 740–748.

https://doi.org/10.1016/j.knosys.2010.12.010