2011

Hierarchical Clustering Algorithm with Dynamic Tree Cut for Data Imputation

L. Dee Miller
University of Nebraska-Lincoln, lmille@cse.unl.edu

Nate Stender
University of Nebraska-Lincoln, nstender@cse.unl.edu

Leen-Kiat Soh
University of Nebraska-Lincoln, lsoh2@unl.edu

Ashok Samal
University of Nebraska-Lincoln, asamal1@unl.edu

Kevin A. Kupzyk
University of Nebraska-Lincoln, kkupzyk2@unl.edu

Hierarchical Clustering Algorithm with Dynamic Tree Cut for Data Imputation

L. Dee Miller, Nate Stender, Leen-Kiat Soh, Ashok Samal
Computer Science and Engineering
University of Nebraska, Lincoln
Lincoln, NE USA
{lmille, nstender, lksoh, samal}@cse.unl.edu

Kevin Kupzyk
Children Youth Families & Schools
University of Nebraska, Lincoln
Lincoln, NE USA
kkupzyk2@unl.edu

***Abstract*—Missing values are very common in real-world datasets for a variety of reasons. Deleting data points with missing values can negatively impact the performance of data analysis methods (e.g., machine learning, data mining). Using a human expert to restore the missing values is expensive and time consuming. The alternative is to impute the missing values during data preprocessing using the known values. This improves performance for data analysis, assuming the imputed values are correct. Unfortunately, imputation algorithms which use all the known values (e.g., mean imputation) often have considerable variance between the imputed and real values. More complex imputation algorithms (e.g., deck and model-based) choose a suitable subset of the data points for imputation. However, a weakness of these algorithms is they use all the variables (i.e., attributes) for imputation even if some of the variables are uncorrelated. Here, we propose a framework called ClustFrame for imputation algorithms that chooses suitable subsets for both data points *and* variables. We also present a ClustImpute algorithm based on our framework that uses single imputation with (1) hierarchical clustering, (2) dynamic tree cut, and (3) a regression model to impute all missing values. Using nine datasets from the UCI repository and an empirically collected complex dataset, we evaluate our algorithm against several existing algorithms including state-of-the-art model-based algorithms that use multiple imputation. Results show that ClustImpute achieves significantly higher imputation accuracy on many of the datasets. We conclude with some suggestions on improvements to our algorithm.**

Keywords- Attribute Clustering, Model-Based Imputation, Deck Imputation, Machine Learning, Data Mining

I. INTRODUCTION

We use the following terminology in this paper: Datasets consist of independent data points with the same set of variables, but generally different values (data points are elsewhere referred to as instances and variables are elsewhere referred to as attributes). We assume that real world data has been converted into datasets because most data analysis methods (e.g., machine learning, data mining, statistics, etc.) are designed to operate on datasets.

Datasets often contain variables with some missing values. McKnight et al. [1] list five common causes for missing data: (1) research design, (2) processing of information, (3) measurement characteristics of equipment used, (4) conditions during data collection, and (5) chance from odd circumstances. Missing values in the dataset negatively impact the performance of data analysis methods. For example, missing values reduce the accuracy of machine learning classifiers [2]. However, using a human expert to correct all the missing values is both expensive and time consuming. Further, a human expert could accidently introduce inconsistencies into the data (i.e., experimenter's bias [3]). Also, excluding all data points with missing values will often leave too few data points for stable analytic results [4]. The solution is to develop algorithms to automatically correct (i.e., impute) all the missing values during data preprocessing. Such algorithms could impute the missing values with less expense and time compared to a human expert. Further, they would be less prone to introducing bias because they always use a consistent methodology.

Note that there are three different kinds of missing values [4]: (1) missing completely at random (MCAR) where the missing value has no connection to the variables, (2) missing at random (MAR) where the cause for missing values is values in other variables, and (3) non-ignorable (NI) where the cause for missing values is in the same variable. In this paper, we focus on MCAR which is used most often for evaluating imputation algorithms [5].

Mean imputation [1] is a widely used imputation approach due to its simplicity. This algorithm imputes each missing value using the mean for that variable in ***all*** the other data points. Unfortunately, there is often considerable variance between imputed values from mean imputation and real values. Additionally, assigning the same values to data points with otherwise different variables makes them less useful for data analysis. For example, assigning the same value to data points with different labels provides no benefit for machine learning classifiers. The solution is to use imputation algorithms which only choose ***suitable*** values. Using only suitable values, such algorithms could impute values that are both more accurate and more useful to the data analysis methods.

There are two commonly used strategies for imputation algorithms which are better able to choose suitable values for imputation: deck- and model-based. Here we provide only a high-level overview as a more detailed description can be found in Section II. Briefly, deck imputation algorithms first choose the data points which are most similar, in terms of known values, to the original data point. Then, it uses the

values in the selected donor data points to impute the missing values. One such approach trains a regression model to predict the values for one variable using the most similar data points [6]. After training, deck imputation predicts the missing values using the regression weights and the known values in the donor data points. Model-based imputation algorithms (e.g., expectation-maximization and imputation-posterior [7]), on the other hand, first create a model for all the variables. Specifically, they estimate the parameters for the multivariate distribution for these variables. Then, they impute the missing values by estimating values from the multivariate distribution (i.e., the model) using the known values in similar data points.

Both the deck and model-based strategies select the data from the complete dataset consisting of all the data points and variables. Both strategies are able to choose the subset of data points which are the most suitable for imputation. Generally, this subset contains the data points with the most similar known values to original data point. However, neither strategy focuses on choosing the subset of variables which are the most suitable for imputation. Instead, both strategies assume that all the variables can be used together for the imputation. However, it is known that including irrelevant variables negatively impacts deck imputation [4] and that using independent variables violates a key assumption in model-based imputation [8]. Therefore, there is a need for an imputation algorithm which can choose both the subset of data points and the subset of variables used to impute the missing values.

Our proposed imputation algorithm, called ClustImpute, uses a clustering approach to select suitable subsets as part of the imputation process. First, our algorithm creates separate clusters for both the variables and the data points. Then, we train a separate regression model for each combination of subsets and use these regression models to impute the missing values. Such an approach overcomes the inherent problems with using all the variables together for imputation and allows ClustImpute to choose more suitable values for imputation. This should allow ClustImpute to outperform existing imputation algorithms in correctly imputing the original missing values.

The rest of this paper is organized as follows: Section II gives a more extensive background on existing imputation algorithms including constant replacement, deck imputation and model-based imputation. It also discusses the differences between single and multiple imputation. Section III discusses our proposed framework and the imputation algorithm in more detail. Section IV discusses the imputation algorithms used in the experiments. Section V gives the experimental setup and discusses results, comparing ClustImpute with existing imputation algorithms. Finally, Section VI summarizes the paper and discusses future work on the ClustImpute imputation algorithm.

II. Background & Related Work

In this section, we discuss previous work on data imputation. Due to space considerations, we focus on the categories including imputation methods used in the experiments section below: constant replacement, deck, and model-based. References for all categories and the methods used in the experiments are summarized below in Table 1. Interested readers should consult McKnight et al. [1] and Schafer [7] for a more comprehensive overview on data imputation methods.

A. Constant Replacement Imputation

Constant replacement methods impute each missing value with a constant value [1]. Missing values for one variable are computed using the known values for that variable in the other data points. All missing values are generally replaced with the same constant value. Constant replacement methods include [1]: *mean* imputation, *median* imputation, and *zero* imputation. The difference between the above methods involves how the constant values are computed using, respectively, the mean, median or zero. All of these methods are prone to several problems including underestimating the variance for variables [9], and neglecting correlations between variables, leading to poor imputation results. However, such methods are still widely used because they are easy to implement and classifiers can achieve reasonable accuracy on datasets imputed with these methods [10].

We use mean imputation as the baseline for our experiments. Previous work has shown that the differences between mean and median imputation are minimal [9]. Zero imputation assumes zero is the worst plausible value which is not the case for many of our datasets.

B. Deck Imputation

Deck imputation methods impute each missing value using donor data points [1][4][11]. In *hot* deck imputation, the donors are other data points in the dataset; whereas, in *cold* deck imputation, the donors are selected from another related dataset with similar variables, such as a previous survey taken by the same individuals [1]. We focus exclusively on hot deck imputation because it is much more common and our datasets are generally unrelated, making cold deck imputation infeasible. Basic hot deck methods select donor data points either *randomly* or *deterministically*. For example, a random method randomly chooses a data point with known values and uses its values for imputation [1], while a deterministic method uses a distance metric to choose the data point with the most similar known values [4]. More advanced hot deck methods are generally deterministic or hybrids. For example, Song and Shepperd [4] use a deterministic method involving class mean imputation with *k*-nearest neighbor (MINI) that always chooses the same donor points using only the subset of relevant variables chosen with feature selection. Our proposed clustering method is also deterministic because it always groups together data points with similar values. On the other hand, Gheyas and Smith [8] train a generalized regression neural network using random weights to deterministically choose donor points for imputation and

Siddique and Belin [11] train a regression model using donor points selected randomly based on the inverse distance. The advantage to using hot deck is that it can impute realistic values from the donor data points without the need for strong assumptions on the parametric estimates for the variables [6]. The disadvantage is that it assumes actual values are available in the donor data points, and this assumption may not be valid for datasets with a high percentage of missing values.

The MINI method is the most similar hot deck method to our proposed clustering method. Both first select a specified number of data points using a distance metric, and then MINI uses mean imputation to determine the missing values, while cluster uses a linear regression model. MINI uses *k*-nearest neighbor (kNN) [4] to select the donor points with similar variables based on the labels; whereas our clustering method uses the dynamic tree cut algorithm [4] to select variables from a hierarchical clustering dendrogram created using all the data points. The distance metric for MINI measures similarity using only *variables with known values for the current data point*. Because the other variables are not considered by the distance metric, the donor points selected could have considerable variance in their values for the same variable. As discussed previously, such variance leads to poor imputation results from mean imputation. Our clusters, on the other hand, are created using a distance metric that minimizes the variance *for all known values in the clustered data points*. Further, kNN is limited to data points with discrete labels whereas our clustering method does not require discrete labels. Finally, in datasets with large amounts of missing values, it may be impossible to find the specified number of donor data points (i.e., those with known values) for some variables. In this case, MINI is limited to using fewer data points which could bias the imputation results. Our method uses clustering to select other variables with similar known values and uses these variables to impute the missing values.

In our comparative studies we do not, however, include MINI due to a key difference between the MINI approach and those considered in our experiments: MINI runs feature selection to find the relevant variables based on labels and *only imputes the missing values for these variables*. On the other hand, the other imputation methods (e.g., mean imputation, model-based, and our clustering method) impute values for all the variables (both relevant and irrelevant) and do *not* have access to the labels—a key assumption about the problem domain in our focus. Thus, we do not include MINI in our experiments.

C. Model-Based Imputation

Model-based methods attempt to model the underlying distribution for the datasets. They use the known (observed) values to generate parameter estimates for the underlying multivariate normal distribution for the variables [1][7]. Missing values are then imputed from this distribution using various approaches depending on the estimation method used. Here we briefly discuss two commonly used model-based methods [1][4][7]: (1) *Expectation-Maximization* and (2) *Imputation-Posterior*. Interested readers should consult Schafer [7] for more details on other model-based methods.

Expectation-Maximization (EM) involves two steps which are repeated until a convergence criterion is met. In the Expectation step, missing values are imputed based on the known values and the parameter estimates. Generally, EM uses regression methods to predict the missing values from the known values in other variables [1][4]. In the Maximization step, the parameter estimates are re-estimated using both the known and imputed values. EM converges when the likelihood function for the parameter estimates no longer changes considerably between iterations. The advantage to using EM is that it continues to improve the parameter estimates from one iteration to the next until it reaches convergence. This makes convergence easy to measure compared to IP. The disadvantage is that EM has the potential to converge at local optima [7], meaning the best possible solution may not be reached.

Imputation-Posterior (IP) [7] also involves two steps which are repeated many times until the distribution converges. In the Imputation step, missing values are imputed using random draws from the distribution based on the current parameter estimates. In the Posterior step, the parameters are re-estimated using the known and imputed values. The difference between the IP and EM is in the imputed values. In IP, the imputed values come from the entire distribution whereas in EM they come from deterministic calculations [4]. IP converges after a heuristic determines that the parameter estimates are from the actual distribution. IP is less prone to getting stuck in local optima than EM because of its stochastic approach to imputing values. However, IP is only guaranteed to converge to the actual distribution with an infinite number of iterations. Heuristics may stop IP too early leading to poor imputation results [7].

Model-based methods use an approach quite different from our proposed clustering method. Model-based methods focus on modeling the underlying multivariate distribution for all the variables whereas our algorithm focuses on dividing the dataset into clusters containing the most suitable donor points. Our method is less able to take advantage of multiple imputations (described below) to improve imputation results because the cluster dendrogram is created deterministically. On the other hand, model-based methods require strong assumptions [8] including that the underlying multivariate distribution is approximately normal [7]. Our clustering method is not subject to these assumptions allowing it to achieve improved results on datasets where they do not hold. Finally, model-based methods leverage all the variables into the parameter estimates allowing multiple variables to be used for each missing value. Our clustering method uses subsets of variables to impute the missing values which should improve imputation results when variables in the dataset are

not all jointly normally distributed because independent variables are not being used to impute the missing values.

We use both EM and IP for comparison in our experiments because both give good imputation results [4] but have different advantages/disadvantages. For EM, we use the Amelia method [12] and for IP we use the SAS implementation based on Markov Chain Monte Carlo. Both methods are designed to use *multiple imputation* (MI), where the same method is used to generate multiple imputed values which are averaged to impute each missing value. We use the MI versions because they generally give better imputation results [1] and are more consistent with the state-of-the-art [8].

TABLE I References for Different Imputation Categories and Specific Methods used in Our Experiments

Category	Reference	Method Used
Constant Replacement	Mean [1], Median [1], Zero[1]	MeanSub
Deck	Random [1], Deterministic [4], Hybrid [6][8][11]	ClustImpute
Model-Based	EM & IP [1][7][12][13]	Amelia & SAS

III. METHODOLOGY

In this section, we first we discuss all three components (clustering, dynamic tree cut, and regression) for our proposed ClustImpute imputation algorithm individually. We provide a high level description for each component and we also discuss any parameters which must be specified for that component. Then, we discuss our overall framework ClustFrame showing where each component in ClustImpute fits into the framework. Pseudocode for the ClustImpute imputation algorithm can be found in Fig. 1 at the end of this section.

A. Cluster Components

The first component we use is agglomerative hierarchical clustering [14]. In general, hierarchical clustering algorithms create a tree-like dendrogram containing multiple sets of clusters with different numbers of data points ranging from 1-point clusters at the leaves of the tree to a single cluster containing all the data points at the root. An agglomerative hierarchical clustering algorithm starts with 1-point clusters at the leaves and successively merges each cluster into larger clusters until it has merged all the data points into a single cluster at the root. This is done by merging the clusters containing the most similar data points together. Similarity is measured using a distance metric on the values. Variable clustering is done in the same way except that the variables and data points are transposed.

The second component used is the dynamic tree cut algorithm [15]. Dynamic tree cut (DTC) is used to select the suitable set of clusters from the dendrogram. DTC starts with a very high cut in the dendrogram near the root level. Then, it finds the difference between the list of heights in the dendrogram and the reference height, which is the average of the heights on the list. This results in a list of differences, which necessarily contains some positive and some negative values. The point at which the members of this list cross over from negative to positive is called the breakpoint, which separates two clusters. The number of elements in the list after the breakpoint is called the forward run length. Once the breakpoint is identified and the forward run length is found, the algorithm determines if the resulting cut at the breakpoint would create significant clusters by comparing the forward run length to a threshold parameter. Finally, DTC compares the cluster sizes against the minimum size parameter and merges small clusters with neighboring clusters in the dendrogram.

The third component used in our imputation algorithm is a regression model [2]. Our algorithm trains a separate regression model for each variable containing missing values in the dataset. This model is trained using only the variables in the suitable subsets (i.e., in the same variable clusters) as determined by DTC. The regression model finds a hyperplane such that the distance, based on some distance metric, between the all data points and the hyperplane is minimized. Then, it imputes all the missing values for its variable using the regression weights and the known values for the subset of donor data points with the most similar values. Such donor points often include the original data point assuming its other values are known. Donor points are chosen using the same distance metric originally used to fit the hyperplane.

B. ClustFrame Framework

The ClustImpute imputation algorithm is representative of a larger framework of imputation algorithms which choose the subset of data points and the subset of variables most suitable for imputing the missing values. Our proposed framework consists of three separate steps with the components in ClustImpute on the right-hand side:

Step 1: Choose Variable Subsets → Clustering + DTC
Step 2: Choose Data Point Subsets → Regression
Step 3: Impute Missing Value → Regression

These steps allow ClustFrame to better impute the missing values as discussed in Section I. In Steps 1 and 2, the ClustFrame chooses the subset of variables and the subset of data points. Step 1 is done before Step 2 to allow the most complete (i.e., with all the data points) comparison of the individual variable distributions. Both subsets are used on the dataset to find the most suitable values for imputation. Then, in Step 3, these suitable values are used to impute all the missing values. We can also plug the imputation algorithms described previously (see Section II) into ClustFrame. Mean imputation only uses Step 3 in ClustFrame, whereas existing deck and model-based algorithms only use Steps 2-3 in ClustFrame.

Fig. 1 gives the pseudocode for the ClustImpute algorithm. There are three parameters: the dataset used (D), the minimum variables in each cluster (MinVar) for dynamic tree cut, and the distance metric used for

hierarchical clustering (Dist). In ClustImpute, choosing the variable subsets (Step 1 in ClustFrame) is done by hierarchical clustering and DTC. This corresponds to Lines 1-2 in Fig. 1. Choosing the data point subsets (Step 2) and imputing the missing values (Step 3) is done by computing a separate Regression model for each variable using only the other variables in its cluster. This corresponds to Lines 3-13 in Fig. 1.

The purpose of including ClustFrame is to demonstrate the flexibility of our ClustImpute algorithm. For example, we could drop hierarchical clustering and DTC and use a different clustering algorithm to choose the variable subsets or we could replace the Regression model with a model-based algorithm. As long it uses all three steps, our imputation algorithm should achieve comparable or superior performance to those using fewer steps. We demonstrate this empirically in Section V.

```
// pseudocode for ClustImpute on Dataset D
ClustImpute(D, MinVar, Dist)
1. Dendro ← HierarchicalClustering(D, Dist)
2. VarClusters ← DynamicTreeCut(Dendro, MinVar, Dist)
3. For each VarClust in VarClusters
4.     Subset ← VarClust ∩ Dataset // dataset with all points, but only the
variables in current cluster
5.     For each Var in VarClust
6.         Model ← Regression(Subset-Var)
7.         For each Point in Subset
8.             If Point[Var] is missing
9.                 Point[Var] = Model(Point)
10.            End If
11.        End For
12.    End For
13.End For
```

Figure 1. Pseudocode for the ClustImpute algorithm. The parameters are the dataset (D), the minimum variables for dynamic tree cut (MinVar) and the distance metric for the clustering algorithm (Dist).

IV. Implementation

In this section, we discuss the implementation details for the imputation algorithms and the classifier used in the experiments. Details on the datasets can be found in the Section V.

Our cluster-based imputation algorithm was written entirely in R, a programming language for statistical computing (http://cran.r-project.org/). We first use the hclust library to perform the hierarchical clustering on the dataset with the missing values. We use the Euclidean distance metric for the datasets in Section V because they all contain numeric values. Second, our algorithm uses the dynamicTreeCut [15] library to cut the trees such that each cluster contains the minimum number of variables specified by the parameter. We use the default threshold parameter for DTC and a minimum cluster size of 7. This is based on the average for the datasets (in Section V) containing the fewest variables such that each dataset contains at least one variable cluster. Third, our algorithm trains regression models from the rWeka library [16] using only the data values in the variable clusters. A separate regression model is trained for each variable. The regression models use Euclidean distance because our datasets contain numeric values. Finally, the regression model imputes missing values based on the other variables in donor data points chosen which are similar to the data point with the missing values. Our algorithm returns a dataset with all missing values replaced with imputed values from the regression models.

We used a Java implementation for the mean imputation algorithm. Mean imputation returns a dataset with all missing values replaced with the mean value for that variable in all the other data points.

We used the R implementation for the Amelia EM MI algorithm [12] found in the Amelia library. For each dataset, we produced five imputed datasets using the Amelia algorithm. Then, for each missing data point in the original, we computed the average value of the five corresponding data points in the imputed datasets, and substituted this for the missing value.

The MI procedure (PROC MI) in SAS version 9.2 was used to perform MI using the IP algorithm [4]. For each dataset, five imputations were performed, which resulted in five imputed datasets. Each imputed dataset contained the same known values, but different imputed values. A SAS macro was written, utilizing SAS Integrated Matrix Language (PROC IML), to create the final imputed dataset by computing the average of the five imputed values for each missing data point. The implementation did not distinguish between continuous and dichotomous variables because the correct logistic regression model for each variable is not known. As a result, predicted values for dichotomous variables were near, but never equal to, the real values of the variables (e.g. .9 or 1.2, as opposed to 1 or 2). Thus, for classifier and other dichotomous variables, the average imputed value was rounded to the nearest integer.

We use the Java weka [16] implementation for the artificial neural network, decision tree, and support vector machine classifiers used in Experiment 2. We use the default parameters for all three classifiers.

V. Results

In this section, we start with a summary of the datasets used in the experiments and also discuss the preprocessing necessary for both experiments. Second, we discuss the empirical running times for the imputation algorithms used in the experiments. Third, Experiment 1 compares the accuracy for our single imputation ClustImpute algorithm to that for several commonly used imputation algorithms. For this experiment, accuracy refers to ability to correctly impute the missing value within a specified degree of precision. The purpose of Experiment 1 is to demonstrate that our algorithm achieves significantly higher accuracy against commonly used single imputation algorithms (mean imputation and hot deck) and also against state-of-the-art model-based multiple imputation algorithms based on imputation-posterior and expectation maximization. Fourth, Experiment 2 evaluates the accuracy for three types of machine learning classifiers

trained using the imputed datasets. For this experiment, accuracy refers to the classifier (i.e., generalization) accuracy of the classifiers on independent test sets. The purpose of Experiment 2 is to determine whether there is significant difference between imputation algorithms beyond the scope of imputing the missing values to within a specified degree of accuracy. For example, an imputation algorithm which does not propagate noisy values could achieve lower imputation accuracy, but a classifier could potentially achieve higher classifier accuracy using its imputed dataset. Finally, we provide a high-level summary of the results for both experiments.

A. *Datasets Used and Preprocessing*

We used ten different datasets in our experiments. The first nine datasets are widely-used benchmark datasets from the UCI machine learning repository [17]. These datasets include: the Bupa Liver Disorders (Bupa), Pima Indian Diabetes (Pima), Radar Returns (Ionosphere), the Wisconsin Breast Cancer datasets (Prognostic, Diagnostic), Sonar Mines (Sonar), Vehicle Silhouettes (Vehicle), Wine Recognition (Wine), and Protein Localization Sites (Yeast). The tenth dataset (iLOG) is a real-world dataset created from student interactions with online learning objects [18]. The iLOG dataset was chosen because it contains a wide range of properties which will impact imputation algorithms (e.g., value noise, highly correlated variables, etc.). All of the above datasets contain only variables with numeric values because several imputation algorithms used in our comparisons only work on numeric variables (e.g., model-based algorithms).

The same preprocessing method was used on all ten datasets (*d)* for both experiments. First, each dataset was divided in half to create a separate training and test set. This was done by selecting data points uniformally at random (UAR) without replacement. Next, in the training set, a percentage of the total values based on the missing value parameter (*m)* were nullified, again, to create datasets with values missing completely at random (MCAR). This results in a set of $m \times d$ ***missing*** datasets using the following percentage values:

$m = (5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90)$

For both experiments, we use five different imputation *(i)* algorithms to impute all the missing values resulting in $i \times m \times d$ ***imputed*** datasets.

$i = (MeanSub, Amelia, SAS, Regression, ClustImpute)$

These algorithms include our ClustImpute imputation algorithm (Cluster), Mean Imputation (MeanSub), Amelia EM MI (Amelia), SAS IP MI (SAS), and Regression hot deck (Regression) all of which are described previously.

Both experiments use the same training sets to guarantee a fair comparison. For Experiment 1, we stop after computing the accuracy on the imputed datasets. For Experiment 2, we train classifiers using the ***imputed datasets*** and evaluate the classifiers using the original test sets.

B. *Running Time Comparison*

Here, we discuss the empirical running times for the Amelia, SAS, Regression, and ClustImpute algorithms. MeanSub is not included because it performs only a single calculation for each variable that requires almost no running time. Due to space considerations, Table II contains only the results for three datasets and three missing rates (10%, 40%, and 70%). However, the running time on the Bupa, Diagnostic, and Wine datasets is representative of those on the other datasets considered. The running times in Table II include all the separate runs for the MI algorithms (SAS and Amelia). We observe that the running time for all the imputation algorithms increases with the missing values. This is expected because (1) fewer known values make it more difficult for all algorithms to choose the donor points used to impute the missing values and (2) there are more missing values that need to be imputed. Also, the Regression and ClustImpute single imputation algorithms have much longer running times than the model-based MI algorithms. First, both Regression and ClustImpute create a separate regression model for each variable whereas Amelia and SAS only estimate a single set of parameters for the multivariate distribution. Second, Regression and ClustImpute are implemented in R because the only implementation for dynamic tree cut is written in R. The R programming language is a scripting language that runs slower than other more optimized languages such as Java or the MI function built into SAS. Overall, the Regression and ClustImpute algorithms require longer running times, but result in improved imputation accuracy on some datasets as shown in Experiment 1 below.

TABLE II Running Time in Seconds for the Amelia, SAS, Regression, and Cluster (i.e., ClustImpute) Algorithms on the Bupa, Diagnostic, and Wine Datasets. The running times for the MI algorithms (i.e., Amelia and SAS) include all the separate runs. The "Miss" column indicates the missing rates (i.e., 10 means 10% missing).

Dataset	Miss	Amelia	SAS	Regression	Cluster
bupa	10	1	2	6	15
bupa	40	11	2	21	35
bupa	70	55	3	37	45
diagnostic	10	29	41	68	195
diagnostic	40	137	55	286	345
diagnostic	70	137	55	452	513
wine	10	5	3	7	17
wine	40	28	4	27	36
wine	70	48	5	51	48

C. *Experiment 1: Imputation Accuracy*

In this experiment, we compute the imputation accuracy for all the imputed datasets. A missing value is *correctly imputed* when it falls inside a range around the original value from the train set using half the standard deviation for that variable. This equivalence testing is similar to that described in Wellek [19]. The overall imputation accuracy for one dataset is the number of correctly imputed values over the total number of missing values.

The average imputation accuracy on all datasets is given in Table III. A ✓ indicates that ClustImpute achieves statistically sigificantly higher accuracy than that algorithm (based on a two-tailed *t*-test), while a × indicates the

opposite. The (No Data) entries indicate Amelia did not run on the majority of the missing rate because the number of data points was too low to estimate the parameters. Space consideration prevent us from showing imputation accuracy versus missing percentage trends for all datasets. However, Fig. 2-3 show the trends for two representative datasets: Sonar where ClustImpute achieves higher accuracy and Vehicle where it achieves lower accuracy. Regardless, the results in Table III show that *our ClustImpute algorithm achieves higher accuracy on most of the datasets compared to the existing algorithms*. First, ClustImpute outperforms MeanSub on all datasets. Its combined approach using clustering and regression allows far more precision when imputing missing values than taking the mean value for the entire variable. On the other hand, MeanSub uses only a simple calculation requiring less time than the steps in the ClustImpute framework. Therefore, ClustImpute should be used unless speed is more imporant than accuracy. Second, ClustImpute achieves slightly higher overall performance than the model-based MI approaches (i.e., SAS and Amelia). ClustImpute achieves statistically higher accuracy compared to SAS on Bupa, Pima, Sonar (see Fig. 2), and Yeast and lower accuracy on Diagnostic, Prognostic and Vehicle (see Fig. 3). Results for ClustImpute and Amelia are comparable to those for ClustImpute and SAS on the datasets where Amelia works. After some analysis, we discovered that datasets where SAS significantly outperformed ClustImpute, including Diagnostic, Prognostic, and Vehicle, all contained large numbers of highly correlated variables. The same was true for iLOG where SAS also outperformed ClustImpute. We evaluated iLOG because it contains highly correlated variables which are also somewhat redundant, allowing them to be safely removed without deleting uniquely useful variables. We found that, after the highly correlated variables were removed, the accuracy for SAS dropped significantly with 0.04 lower accuracy averaged over all the missing rates. This is reasonable because model-based algorithms assume all variables are part of underlying multivariate normal distribution [7]. They achieve lower accuracy on datasets containing variables with lower average correlation which violates this assumption consistent with the discussion in Gheyas & Smith [8]. On such datasets, ClustImpute has a significantly higher accuracy trend than does SAS (see Fig. 2) until significant (≥ 50) amounts of missing values make it difficult to train regression models on the variable subsets. On such datasets, ClustImpute is the better choice for imputation. Third, we found ClustImpute achieves comparable or superior performance to Regression. On the iLOG, Ionosphere, and Sonar datasets, ClustImpute achieves significantly higher accuracy. This shows that the clustering improves the results compared to just using Regression hot deck. The datasets with no change are those containing a limited number of variables (Bupa, Pima, Wine, and Yeast) where the variable clustering makes no difference and those containing highly correlated variables (Diagnostic, Prognostic, and Vehicle). Such datasets contain so many highly correlated variables that our algorithm cannot fit them all into the same clusters with fixed size. As a result, some variables which could be used for imputation are unavailable because they are in different subsets. On the Vehicle dataset, containing the largest number of significantly correlated variables, clusters with insufficient size make imputing the correct values more difficult compared to Regression which uses all the variables. This results in the trend where ClustImpute has significantly lower accuracy than Regression (see Fig. 3) until increasing missing values (missing percent ≥ 40) degrade the regression models in both algorithms mitigating the impact using variable subsets. In the future, we intend to modify ClustImpute to dynamically choose the number of variables per subset based on the total number of variables and their average correlation.

TABLE III Average Imputation Accuracy for Cluster (i.e., ClustImpute) and Other Algorithms. A ✓ indicates Cluster achieves significantly higher accuracy that that algorithm, while a × indicates the opposite. (No Data) indicates the algorithm did not run on the datasets.

Dataset	MeanSub	Amelia	SAS	Regression	Cluster
bupa	0.45 ✓	0.41 ✓	0.41 ✓	0.49	0.49
diagnostic	0.40 ✓	0.82 ×	0.82 ×	0.73	0.73
iLOG	0.37 ✓	(No Data)	0.58	0.43 ✓	0.54
ionosphere	0.41 ✓	(No Data)	0.60	0.58 ✓	0.62
pima	0.36 ✓	0.40 ✓	0.42 ✓	0.46	0.46
prognostic	0.40 ✓	(No Data)	0.68 ×	0.61	0.61
sonar	0.35 ✓	(No Data)	0.51 ✓	0.44 ✓	0.55
vehicle	0.36 ✓	0.78 ×	0.78 ×	0.71 ×	0.65
wine	0.33 ✓	(No Data)	0.45	0.47	0.47
yeast	0.54 ✓	0.39 ✓	0.46 ✓	0.56	0.57

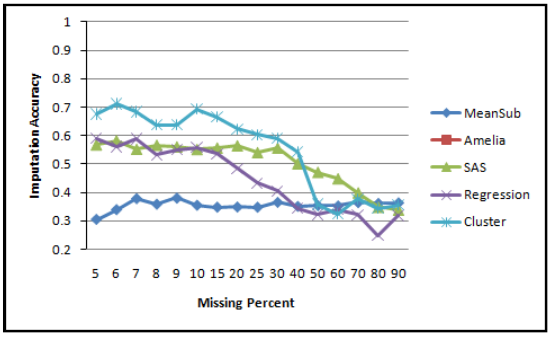

Figure 2. Imputation Accuracy Trend for all Algorithms on the Sonar Dataset. Cluster (i.e., ClustImpute) achieves significantly higher accuracy than all the other algorithms on this dataset.

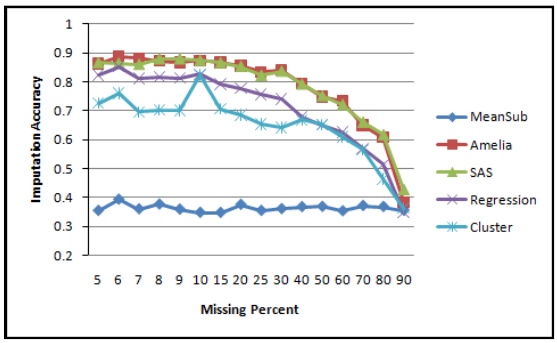

Figure 3. Imputation Accuracy Trend for all Algorithms on the Vehicle Dataset. Cluster (i.e., ClustImpute) achieves significantly lower accuracy than all the other algorithms on this dataset except mean imputation.

D. Experiment 2: Classifier Accuracy

The quality of data imputation methods cannot be measured purely on their accuracy. The ultimate measure of performance is how well they benefit the data analysis methods. Here, we evaluate whether the imputation algorithm used has a significant impact on the classifier accuracy of three commonly used supervised learning classifiers. It is possible that an imputation algorithm with lower overall accuracy could still result in higher classifier accuracy, which is measured on an independent test set. For example, an imputation algorithm with lower imputation accuracy could benefit the classifier by reducing the impact of noisy values.

In this experiment, we evaluate three separate classifiers trained on the imputed datasets: support vector machine (SVM), artificial neural network (ANN) and decision tree (Tree). All three classifiers use the default parameters from the weka machine learning library [20]. We consider the classification accuracy (i.e., ability to predict correct labels) for all three classifiers on the test set.

The average classification accuracy using the training sets imputed with different algorithms for all datasets is given in Table IV. A ✓ indicates that using ClustImpute to impute the training set resulted in significantly higher classifiction accuracy than using another (based on a two-tailed *t*-test), while a × indicates the opposite. The (No Data) entries indicate Amelia failed to run because the number of data points was too low to estimate the parameters. The results in Table IV show that, in general, *the imputation algorithm used has little impact on the classification accuracy* with one exception: datasets imputed with ClustImpute generally achieve higher classification accuracy on SVM, ANN and Tree than those imputed with MeanSub. This is reasonable because MeanSub replaces all missing values (even those for data points with different labels) with the same imputed value. Such values are no longer useful for training the classifier because they cannot be used to separate data points with different labels. Only on the Prognostic dataset for the Tree classifier does MeanSub benefit the classifier. The Prognostic dataset contains large amounts of noisy values making precise classification difficult. On such a dataset, ClustImpute recovers the noisy values resulting in lower classification accuracy compared to MeanSub which renders them inert. However, we only notice a difference on the Tree classifier which does not use an iterative training process to compensate for the inert values from MeanSub by focusing even more on the known values which still contain some noise. Regardless, using MeanSub should generally be discouraged on datasets which will be used to train classifiers. Otherwise, the classification accuracy seems to correspond to the imputation accuracy on the dataset. Thus, the decision on what imputation algorithm to use depends more on the variables as discussed in Experiment 1.

TABLE IV Average Classifier Accuracy Using Training Sets Imputed with all Algorithms. A ✓ indicates the Classifier achieves significantly higher accuracy using Cluster (i.e., ClustImpute), while a × indicates the opposite. (No Data) indicates the algorithm did not run on the datasets.

Classifier	Dataset	MeanSub	Amelia	SAS	Regression	Cluster
ANN	bupa	0.63	0.59	0.60	0.62	0.62
ANN	diagnostic	0.90 ✓	0.94	0.93	0.93	0.93
ANN	iLOG	0.68	(No Data)	0.66	0.65	0.66
ANN	ionosphere	0.78	(No Data)	0.80	0.78	0.79
ANN	pima	0.71	0.72	0.72	0.72	0.72
ANN	prognostic	0.70	(No Data)	0.69	0.68	0.71
ANN	sonar	0.64	(No Data)	0.68	0.64	0.65
ANN	vehicle	0.56 ✓	0.65	0.64	0.65	0.62
ANN	wine	0.78	(No Data)	0.84	0.81	0.81
ANN	yeast	0.42 ✓	0.46	0.46	0.48	0.48
SVM	bupa	0.60	0.57	0.58	0.60	0.60
SVM	diagnostic	0.87 ✓	0.97	0.96	0.94	0.95
SVM	iLOG	0.67	(No Data)	0.66	0.67	0.66
SVM	ionosphere	0.77 ✓	(No Data)	0.82	0.78	0.80
SVM	pima	0.70 ✓	0.74	0.74	0.74	0.74
SVM	prognostic	0.78	(No Data)	0.74	0.76	0.76
SVM	sonar	0.62	(No Data)	0.68	0.65	0.65
SVM	vehicle	0.45 ✓	0.57 ×	0.57 ×	0.54	0.52
SVM	wine	0.74	(No Data)	0.88 ×	0.82	0.81
SVM	yeast	0.35 ✓	0.42	0.42	0.44	0.44
Tree	bupa	0.62	0.61	0.60	0.62	0.62
Tree	diagnostic	0.88 ✓	0.93	0.92	0.91	0.91
Tree	iLOG	0.62	(No Data)	0.62	0.62	0.64
Tree	ionosphere	0.79	(No Data)	0.83	0.81	0.82
Tree	pima	0.69	0.71	0.71	0.71	0.71
Tree	prognostic	0.74 ×	(No Data)	0.69	0.68	0.67
Tree	sonar	0.62	(No Data)	0.65	0.65	0.63
Tree	vehicle	0.52 ✓	0.58	0.59	0.57	0.56
Tree	wine	0.69 ✓	(No Data)	0.79	0.77	0.76
Tree	yeast	0.39 ✓	0.41	0.41	0.44	0.44

E. Experiment Summary

Here, we summarize the results for both our experiments comparing our proposed imputation algorithm with existing imputation algorithms. For Experiment 1, as expected, the algorithms using subsets of data points achieved superior imputation accuracy to mean imputation on all datasets. Our ClustImpute single imputation algorithm, which uses both subsets of variables and data points, achieved superior imputation accuracy to model-based, multiple imputation algorithms (i.e., Amelia EM MI and SAS IP MI) on many datasets. ClustImpute also achieves superior accuracy to single imputation Regression used as a component. However, model-based imputation algorithm still achieves superior accuracy on several datasets containing numerous, highly-correlated variables. It is easier for model-based algorithms to fit underlying multivariate distribution on such datasets. Also, the fixed minimum cluster size in our algorithm results in correlated variables being assigned to different cluster subsets. Therefore, we recommend our ClustImpute imputation algorithm for any dataset without numerous, highly-correlated variables. For Experiment 2, we found that the imputation algorithm used has little impact on the classifier accuracy. The one notable exception is mean imputation, which causes a significant drop in classification accuracy on several datasets. Mean imputation imputes the same missing values for all data points including those with different labels. This makes it more difficult for

the classifier to separate data points based on their labels. Therefore, we recommend avoiding mean imputation on datasets used for classification and otherwise following the above suggestions for choosing the imputation algorithm.

VI. Conclusions & Future Work

In this section, we summarize our paper and discuss future work on ClustImpute imputation algorithm.

A. Conclusions

Datasets often contain variables with missing values. Such missing values could be caused by (1) research design, (2) processing of information, (3) measurement characteristics, (4) conditions during data collection, and (5) chance from odd circumstances [1]. Simply excluding data points with missing values can negatively impact the results from the data analysis methods. The alternative is to use data imputation algorithms to correct the missing values using the known values. Using all the known values at once (e.g., mean imputation) can result in substantial bias between the imputed and missing values. Thus, there has been considerable work on developing imputation algorithms to select only suitable values. There are two main strategies: deck and model-based. Deck algorithms first choose donor data points and then use the values from just the donors to impute the missing values. Deck algorithms include neural networks [8], *k*-nearest neighbor [4] and regression models [6]. Model-based algorithms, on the other hand, first model the underlying multivariate distribution for all the variables and then impute the missing values using data points with similar known values. Model-based methods include expectation-maximization [12] and imputation-posterior [7]. Both strategies are interested in choosing suitable data points, but neither is concerned with choosing suitable variables. The failure to do so can negatively impact the imputation accuracy for both strategies. We discuss a framework for imputation algorithms which does both.

We propose a novel ClustImpute imputation algorithm based on our framework which uses (1) hierarchical clustering, (2) a dynamic tree cut algorithm [15], (3) and a regression model to leverage both subsets of variables and subsets of data points for data imputation. We compare our ClustImpute imputation algorithm against four other algorithms including two state-of-the-art algorithms using multiple imputation. These four algorithms are (1) mean imputation, (2) Amelia EM MI, (3) SAS IP MI, and (4) a basic Regression model. The imputation algorithms are all compared on ten datasets over two separate experiments. Overall, our results show that ClustImpute achieves comparable to superior imputation accuracy against all other imputation algorithms considered. Specifically, ClustImpute, using single imputation, outperforms state-of-the-art multiple imputation algorithms except on datasets with a large number of highly correlated variables. We also show that the imputation algorithm used has little impact on classifier accuracy for machine learning classifiers.

B. Future Work

When using the dynamic tree cut algorithm to create our clusters we saw that there was a potential to increase the effectiveness of our method by finding a way to better select the minimum cluster size. For the experiments conducted thus far we have manually selected a minimum cluster size, and, although we noticed a difference in the performance of the algorithm when the number was changed, the results were inconsistent for the various datasets. This seems to imply that there is a unique optimal minimum cluster size for different datasets, and it would certainly be worth investigating the validity of this notion, and determining a way to select this optimal minimum cluster size. Instead of selecting a static number to use for all datasets, we might better select the minimum cluster size as a function of the dataset size, or perhaps develop a completely separate method for selecting the most appropriate number.

An alternative answer to the problem of selecting the proper minimum cluster size is to remove the need to specify one. Instead, it may be beneficial to make use of a hybrid tree cut [15], an algorithm similar to dynamic tree cut which uses a different cut criterion. Hybrid tree cut focuses more on creating well shaped clusters, aiming to create clusters with dense cores of tightly packed nodes and few outliers. Potentially, this distinction in cluster creation could lead to clusters in which variables are more closely related and thus more useful for prediction. Thus, we intend to rerun our ClustImpute imputation algorithm with hybrid tree cut instead of dynamic tree cut.

The missing data in this study was missing completely at random (MCAR). A future study will be conducted to determine if the results observed here generalize to a situation where data are missing at random (MAR). If the cause of missingness is contained in other variables in the dataset, methods that can capitalize on that information should perform better than those that do not consider correlations between variables.

Another possibility worth exploring which could influence the effectiveness of our algorithm is the selection of our distance function. Because the distance matrix is pivotal in the creation of the dendrogram and thus the resulting clusters, selecting the best possible distance function will have an important effect on the results. Thus, we intend to investigate the accuracy of the clustering algorithm with different distance metrics.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 0632642 and an NSF GAANN fellowship.

References

[1] P.E. McKnight, K. McKnight, S. Sidani, and A.J. Figueredo, *Missing data*, Guilford Press, 2007.

[2] S.B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Proceeding of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, IOS Press, 2007, pp. 3-24.

[3] D.L. Sackett, "Bias in analytic research," *Journal of Chronic Diseases*, vol. 32, 1979, pp. 51-63.

[4] Q. Song and M. Shepperd, "A new imputation method for small software project data sets," *Journal of Systems and Software*, vol. 80, 2007, pp. 51-62.

[5] M. Saar-tsechansky, F. Provost, and R. Caruana, "Handling missing values when applying classification models," *Journal of Machine Learning Research*, 2007, pp. 1625-1657.

[6] R.R. Andridge and R.J.A. Little, "A Review of Hot Deck Imputation for Survey Non-response," *International Statistical Review*, vol. 78, 2010, pp. 40-64.

[7] J. Schafer, *Analysis of Incomplete Multivariate Data*, Chapman and Hall/CRC, 1997.

[8] I. Gheyas and L. Smith, "A Novel Nonparametric Multiple Imputation Algorithm for Estimating Missing Data," *Proceedings of the World Congress on Engineering*, 2009.

[9] E. Acuna and C. Rodriguez, "The treatment of missing values and its effect in the classifier accuracy," *Classification, Clustering and Data Mining Applications*, Springer-Verlag, 2004, pp. 639-648.

[10] D.J. Mundfrom and A. Whitcomb, "Imputing Missing Values: The Effect on the Accuracy of Classification.," 1998.

[11] J. Siddique and T.R. Belin, "Multiple imputation using an iterative hot-deck with distance-based donor selection," *Statistics in Medicine*, vol. 27, 2008, pp. 83-102.

[12] G. King, J. Honaker, A. Joseph, and K. Scheve, "Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation," *American Political Science Review*, vol. 95, 2001, pp. 49-69.

[13] T. Lin, "A comparison of multiple imputation with EM algorithm and MCMC method for quality of life missing data," *Quality and Quantity*, vol. 44, Feb. 2010, pp. 277-287.

[14] D.J. Hand, H. Mannila, and Padhraic Smyth, *Principles of Data Mining*, The MIT Press, 2001.

[15] P. Langfelder, B. Zhang, and S. Horvath, "Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R," *Bioinformatics*, vol. 24, Mar. 2008, pp. 719-720.

[16] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*, Morgan Kaufmann, 2005.

[17] A. Asuncion and D. Newman, *UCI Machine Learning Repository*, University of California, Irvine, 2007.

[18] S. Riley, L.D. Miller, L. Soh, A. Samal, and G. Nugent, "Intelligent Learning Object Guide (iLOG): A Framework for Automatic Empirically-Based Metadata Generation," *Artificial Intelligence in Education*, 2009, pp. 515-522.

[19] S. Wellek, *Testing Statistical Hypotheses of Equivalence*, Chapman and Hall/CRC, 2002.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, *The WEKA Data Mining Software: An Update*, SIGKDD Explorations, 2009.