Using Classifier-Based Nominal Imputation to Improve Machine Learning

Xiaoyuan Su[1], Russell Greiner[2], Taghi M. Khoshgoftaar[1], and Amri Napolitano[1]

[1] Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA

[2] Department of Computing Science, University of Alberta, Edmonton, AB, Canada T6G 2E8

suxiaoyuan@gmail.com, rgreiner@ualberta.ca, taghi@cse.fau.edu, amrifau@gmail.com

Abstract. Many learning algorithms perform poorly when the training data are incomplete. One standard approach involves first imputing the missing values, then giving the completed data to the learning algorithm. However, this is especially problematic when the features are nominal. This work presents "classifier-based nominal imputation" (CNI), an easy-to-implement and effective nominal imputation technique that views nominal imputation as classification: it learns a classifier for each feature (that maps the other features of an instance to the predicted value of that feature), then uses that classifier to predict the missing values of that feature. Our empirical results show that learners that preprocess their incomplete training data using CNI using support vector machine or decision tree learners have significantly higher predictive accuracy than learners that (1) do not use preprocessing, (2) use baseline imputation techniques, or (3) use this CNI preprocessor with other classification algorithms. This improvement is especially apparent when the base learner is instance-based. CNI is also found helpful for other base learners, such as naïve Bayes and decision tree, on incomplete nominal data.

Keywords: incomplete data, imputation, support vector machine, instance-based learning, nominal data.

1 Introduction

It is often difficult to learn good classifiers when the training data are missing attribute values. To deal with missing data in classification tasks, many learners first use some imputation technique to fill in the missing values, before giving the completed data to a complete-data learner. A simple imputation technique is to replace each missing value of a real-valued attribute with the mean of the observed values of the attribute (MEI), or a nominal attribute with its most commonly observed value (MCI). This is used by the WEKA implementations for many classification algorithms [4]. However, these trivial imputers generally do not help produce high-quality classifiers for incomplete data.

Many learning algorithms have algorithm-specific built-in schemes for handling missing value – e.g., naïve Bayes can simply ignore the missing attributes at both learning and classification time, and some instance-based algorithms simply set the distance measure to any attribute (of an instance) missing an entry to the associated maximum value [1]. However, these simple missing data handling methods often do not produce accurate estimates to fill in the missing values. As we anticipate that answers based on accurately imputed data will be better than those based on the original incomplete data and on less accurately imputed data, we expect preprocessing the incomplete data with accurate imputers can boost the classification performance of machine learners on incomplete data. In general, an "imputation-helped learner" uses some imputation techniques to fill in the missing values before training a classifier. Su *et al.* investigated such imputation-helped learners in the context of numeric features, and showed that certain numerical imputation techniques can improve classification performance [9]. However, few existing works specifically investigate how imputation techniques can improve classification performance on nominal data.

This work explores imputation-helped learners for nominal features. We first propose an easy-to-implement algorithm for imputing nominal features, *classifier-based nominal imputation (CNI),* which treats imputation as a classification task: first learn a classifier for each feature, then use this trained classifier to impute values for the missing entries. We let the notation "kNN-CNI(SVM)" refer to the learning system that preprocesses the incomplete training data by learning a SVM classifier for each attribute to impute each missing value of this attribute (so if there are n attributes with missing values, this produces n different classifiers); the resulting completed dataset from this CNI(SVM) preprocessing is then given to the learner kNN ("k nearest neighbors" [1]) to produce a final classifier. In general, the "B-CNI(L)" learner first uses the learner L to learn n different classifiers for imputing values for the n attributes, then gives the completed data to the base learner B.

We first investigate the imputation performance of our proposed CNI imputation using 10 machine learners as imputation classifiers, and found that (1) each of these CNI imputers has more accurate predictions than the baseline kNN imputation and MCI, and (2) CNI(SVM) (classifier-based nominal imputation that uses support vector machine) and CNI(DT) (that uses decision tree) perform especially well.

By applying the top-performing CNI imputers to preprocess incomplete nominal data, our empirical experiments show that these imputation techniques can significantly improve classification performance for instance-based algorithms on incomplete nominal data with either a high or low percentage of missing values. While imputation is not as critical for other learning algorithms, such as naïve Bayes and decision tree, we found that our proposed B-CNI(L) approach can still boost their classification performance when the missing ratio is at or below 20%.

Section 2 describes the framework of this paper, Section 3 provides our experimental design and results, and Section 4 contains the conclusions.

2 Framework

2.1 Imputation for Nominal Data

As many real-world datasets are missing some information, imputation techniques are often used to fill in the missing values with estimated values; this often leads to performance that is better than just using the original incomplete data. Imputation techniques for numeric data, such as EM (expectation maximization) and BMI (Bayesian multiple imputation), involve iteratively updating estimates of means and covariance matrices, as the data is assumed to be normally distributed [9]. This is, of course, typically not appropriate for nominal data.

A baseline imputation for such nominal data is MCI (most common [value] imputation), which fills the missing values with the most frequently observed value of the attribute. However, MCI distorts the distribution of the values by overestimating the most frequent value, which often leads to incorrect inferences. Figure 1 shows that MCI does not produce a similar shape of attribute value distribution as the original missing data, while our proposed CNI imputation does; see details in the next subsection.

Another well-known nominal imputation technique is kNN imputation (kNNI) [2], which imputes a missing value of an attribute in an instance as the most common value of that attribute in the instance's k nearest neighbors. However, kNNI is not very effective because of the way that kNN selects the nearest neighbors: this is based on a distance function that is problematic in the presence of incomplete data (see Section 2.3 below).

We therefore propose an easy-to-implement nominal data imputation technique: classifier-based nominal imputation. This paper investigates whether this nominal imputation technique can be used to improve classification performance for machine learned classifiers on incomplete nominal data.

2.2 Classifier-based Nominal Imputation

The basic idea of classifier-based nominal imputation (CNI) is simple: treat imputation as classification. For each attribute f_i with missing values, learn a classifier $c_i(\ldots)$ that takes as input the values of the other $n-1$ attributes $\{f_j | j \neq i\}$ for an instance, and returns the value for f_i for this instance. CNI(L)

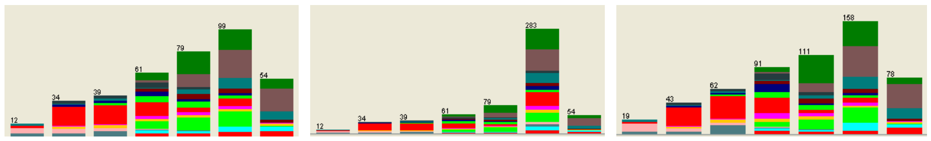

Fig. 1: Attribute value distributions on the "date" attribute of the Soybean-large dataset with 30% of the values missing: (from left to right) (1) the original missing data, (2) after most common imputation (MCI), (3) after CNI using decision tree imputation CNI(DT).

uses the learning algorithm L to learn this $c_i(\ldots)$ classifier from the training instances with observed values on f_i; CNI then uses this $c_i(\ldots)$ to impute the missing values of this attribute in the remaining instances. Algorithm 1 illustrates the CNI imputation algorithm.

After imputing incomplete data using CNI(L), B-CNI(L) passes the resulting completed dataset to the base learner B, which produces a classifier that can then classify (possibly incomplete) test instances. Note that the L learner must be able to deal with incomplete data as its data sample D_i^+ (see Algorithm 1) will typically have missing values, and the $c_i(\ldots)$ classifier that it produces must also be able to handle missing information. However the base learner B will only need to deal with complete instances. The resulting classifier will predict class labels for the original incomplete test data.

Note that the values imputed for one attribute could be used in the later iterations of imputation. For example, if attribute f_1 was the first imputed attribute, its imputed values could be used when imputing values for f_2 and for all other subsequent attributes. However, to simplify the algorithm and focus on the general techniques, we did not use the previously imputed values for any of the later attributes in this work.

Here, we investigate the following Imputation Learners L: decision tree (C4.5), decision table (dTable), lazy Bayesian rules (LBR), naive Bayes (NB), one rule (OneR), decision list (DecList), random forest (RF), support vector machine (SVM), radial basis function neural networks (RBF), and multilayer perceptron neural network (MLP). Each of the learners and the CNI technique are implemented within the WEKA [4] framework, and use WEKA's default approach for dealing with missing values, at both learning and performance time. When there are too few observed feature values to train reasonable classifiers (here, under three instances with observed values for attribute f_i in D_i^+), we simply use MCI to impute a value for that attribute. For comparison, we also implement the baseline nominal imputation techniques kNNI and MCI.

Algorithm 1 CNI(ImputationLearner L)

```
for $t = 1 \ldots n$ do          % over each feature (attribute column of dataset $D$)
    % View *feature* $f_i$ as the class and the other columns as features
    Divide instances $D = D_i^+ + D_i^-$
        where training set $D_i^+$ contains the instances with observed values of attribute
        $f_i$, and test set $D_i^-$ contains the instances that miss values of attribute $f_i$
    Let $c_i(\ldots) = L(D_i^+)$ be classifier trained using learner $L$ on $D_i^+$
        using $L$'s default approach to handle missing data, as necessary
    for each instance $d \in D_i^-$ do
        Let $d_{-i}$ be the $n-1$ "non-$f_i$" values in $d$
        Impute the "$f_i$" value of $d$ as $c_i(d_{-i})$
    % Move on to the next attribute
% return the resulting completed dataset $D'$
```

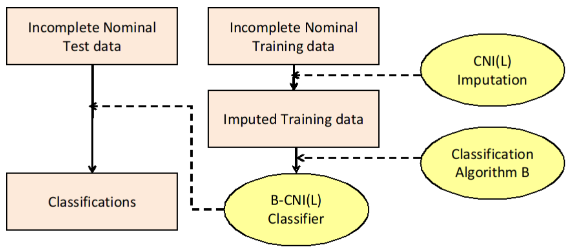
Incomplete Nominal Test data
Incomplete Nominal Training data
CNI(L) Imputation
Imputed Training data
Classification Algorithm B
Classifications
B-CNI(L) Classifier

Fig. 2: The B-CNI(L) framework of improving the classification performance of machine learning algorithm B using L imputation as the preprocessor for the nominal training data.

As these classifiers are well-known, we do not provide detailed descriptions for each of them here. We use default parameter settings of WEKA for L and B learners unless they are explicitly specified otherwise. For the imputation learner L=SVM, we use the linear kernel [7].

2.3 Using CNI to Improve Classification Performance of Machine Learned Classifiers

Figure 2 shows the general B-CNI(L) framework, which first uses some nominal learner L for imputing the missing nominal values in the training set to generate an imputed training dataset, then gives the completed dataset to a learning algorithm B (e.g., kNN) to learn a classifier. We then use this classifier to classify the (possibly incomplete) test instances.

Although it is legitimate to impute training data together with test data (excluding the labels of the test data), our imputers only work on the training data, and use the original incomplete test data for evaluation. (When predicting class labels for each incomplete test instance, we use the missing data handling strategy of the classifier associated with the base learner B). We use this imputation scenario for dealing with incomplete training/test data because, in practice, training data and test data often come in different times. Therefore, it would be impractical to impute training data and test data together in many cases.

We investigate how our proposed CNI imputation can help machine learners such as instance-based learning algorithms, naïve Bayes, decision tree, and neural network.

Instance-based learning is a kind of “lazy learning” that assigns a label to a new unlabeled instance based on the “closest” labeled instances appearing in the training set. A commonly used instance-based learning algorithm is the k-nearest neighbor algorithm, which first identifies the k neighbors nearest to the “to be labeled instance” (based on the distance values calculated between this new instance with each of the instances in the training set) and returns the majority class over these neighbors, or perhaps some variant that weights the labels of these neighbors. The distance function in instance-based learning is usually defined as [1]:

$$Dis(x,y) = \sqrt{\sum_{i=1}^{n} f(x,y)} \tag{1}$$

where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are instances, each over n attributes. For numeric attributes, $f(x_i, y_i) = (x_i - y_i)^2$, and for Boolean and nominal attributes, $f(x_i, y_i) = I(x_i \neq y_i)$ which is 0 if $x_i = y_i$ and is 1 otherwise. The nearest neighbor for an instance x is $argmin_y Dis(x,y)$ over all instances y in the training set.

Of course, Equation 1 is not defined if any $f(x_i, y_i)$ value is undefined. However, as $f(\ldots)$ must return an answer even if either x_i or y_i is missing, many nearest-neighbor systems will use extreme values – ie, set $f(x_i, y_i)$ to the maximum value if either x_i or y_i is missing [1]. As this simple approach often leads to biased distance values, we suspect that a reasonable estimate of missing values in the training set will improve the resulting classifier.

The best value of k often depends on the dataset. In general, a larger k value reduces noise on the classification, but can impair performance if it is too big. A good k can be selected by using various techniques, such as cross-validation or some heuristic. We often consider $k > 1$ nearest neighbors and set k to an odd value to prevent ties for binary classification. (For multiple classes, we use the plurality of the votes.) Some instance-based classifiers will also weight each neighbor in this vote by using $1/Distance$ or $1 - Distance$, both of which give higher weights to nearer neighbors [4].

When using imputation to help instance-based algorithms, imputing the training set will replace the missing values for the instances in the training set, while the missing values that have penalized distance values in the test instances are still present. Imputation reduces the number of times where instance-based learning algorithms will use the maximized distance values.

Many other learning algorithms deal naturally with incomplete data, and therefore imputation may not be so critically helpful. For example, naïve Bayes makes classifications based on only observed values [6], while C4.5, a well-known decision tree classifier, will in effect simply disregard the missing values during training [8].

3 Experimental Design and Results

We explored 12 nominal datasets from the UCI machine learning repository (see dataset description in Table 1) [3]. Six of the 12 datasets have both nominal and numeric attributes; these have *italicized* names in Table 1 – e.g., the dataset “Australian” has 8 nominal attributes out of a total of 14 attributes. Most of the nominal data have more than two attribute values. Here, when corrupting the data by removing values, we only remove some of the nominal attributes, and leave all of the numeric attributes.

To investigate the performance on datasets with different missing ratios, we generate five incomplete datasets for each of the above datasets by randomly deleting 10%, 20%, 30%, 40%, and 50% of the observed values – this is done by removing each [instance, attribute] independently, with probably 0.1 [resp., 0.2, ..., 0.5].

We first evaluate our proposed CNI(L) imputation algorithms using different machine learned classifiers L; we then pick the top two CNI imputers in terms of their estimation accuracy (see next subsection) to impute the incomplete nominal training sets, before applying a base learner to train a classifier on the imputed training set, which we subsequently use to classify the incomplete instances in the test set. The classification results reported are the average of five cross validation folds.

We then evaluate how CNI imputation can improve the classification performance for machine learners kNN, naïve Bayes, decision tree, and MLP on incomplete nominal data.

3.1 Evaluation of CNI Imputation Algorithms

As we generated the incomplete datasets by removing values from the complete datasets, we have the ground truth for each missing value. Therefore, our nominal imputation algorithms can be evaluated in terms of the estimation accuracy by checking the imputed values against their respective ground truth values.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} I(P_i = R_i) \qquad (2)$$

where P_i is the estimated value produced by the imputer (for some [instance, attribute] pair), R_i is the ground truth value, and N is the total number of the imputed values.

Figure 3 illustrates the imputation performance of the CNI imputers $L \in$ {SVM, C4.5, NB, RF}, kNNI and MCI, when 30% of the data is missing.

Table 1: Datasets from the UCI Machine Learning Repository

dataset	#instances	#attributes (nominal/all)	average attribute values	#classes
Chess	3196	35	2.03	2
Corral	128	6	2	2
Flare	1066	10	3.3	2
Mofn-3-7-10	1324	10	2	2
Soybean-large	562	35	2.86	19
Vote	435	16	3	2
Australian	690	8/14	4.63	2
Cleve	296	7/13	2.71	2
Crx	653	9/15	4.56	2
German	1000	13/20	4.31	2
Hepatitis	80	13/19	2	2
Lymphography	148	15/18	2.93	4

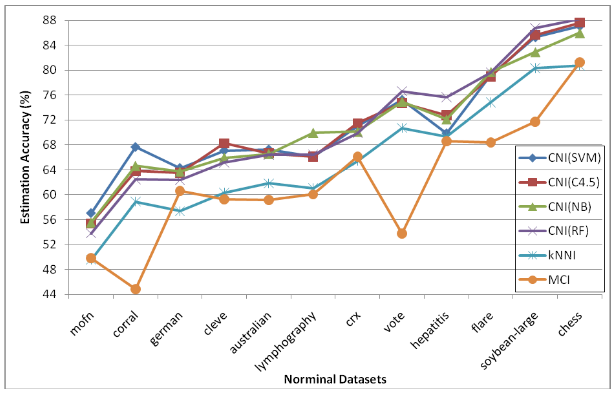

Fig. 3: Estimation accuracies of the CNI imputers, kNNI, and MCI on the datasets with a missing ratio of 30%.

Our proposed nominal imputation algorithm – classifier-based nominal imputation (CNI) – performs significantly better than the commonly used nominal imputers, kNNI and MCI. While each CNI(L) imputer (that uses the learner L) we investigated outperforms kNNI and MCI, each of the top performers CNI(SVM) and CNI(C4.5) has more than 7% and 15% higher average estimation accuracies over kNNI and MCI respectively. Statistically, CNI(SVM) and CNI(C4.5) outperform kNNI with 1-sided t-test $p < 2 \times 10^{-8}$ and $p < 1 \times 10^{-8}$ respectively, and even the worst performing CNI(RBF) and CNI(MLP) still slightly outperform kNNI, albeit with $p < 0.01$ and $p < 0.1$. As expected, the estimation accuracies of the imputers decrease as the missing ratio of the datasets increases.

For all of the 10 nominal imputers we investigated, the ranking of the nominal imputers in terms of their average estimation accuracies over the 12×5 datasets (five datasets with different missing ratios between 10% and 50% generated from each of 12 base datasets) is: CNI(SVM), CNI(C4.5), CNI(DecList), CNI(NB), CNI(LBR), CNI(RF), CNI(OneR), CNI(dTable), CNI(RBF), CNI(MLP), kNNI, and MCI. We will therefore use the best two, CNI(SVM) and CNI(C4.5), as nominal imputers in the investigations below.

3.2 The Impact of Nominal Imputers on the Classification Performance for Instance-based Learning Algorithms

Now we evaluate how much nominal imputers can improve the classification performance for instance-based learning algorithms.

We work on instance-based algorithms with different weighting schemes. Rather than setting the best k value for each different dataset, different nominal imputer and distance weighting scheme, we will instead simply use $k = 5$ for all cases.

Table 2 is a summary of the classification results, with cells that record the average classification accuracy over the 12 nominal datasets. Figure 4(a) illustrates how nominal imputers improve the classification performance of instance-based algorithms on the dataset "Australian" over various different missing ratios.

Table 2 and Figure 4(a) show that instance-based algorithms decrease their classification accuracy fast with the increase of the missing ratio of data. With the help of nominal imputers, kNN can achieve a statistically significant improvement of classification accuracy (here, this is based on a 1-sided paired t-test, with $p < 0.05$; note all statistical claims are based on this test). This is true for all the datasets with all missing ratios we investigated (in the range of 10% and 50%), and for different distance weighting schemes of instance-based algorithms, including (1-distance) weighting, inverse distance weighting, and no weighting.

We anticipate a more accurate imputation technique will further improve the classification performance. Over the 12 datasets, kNN-CNI(SVM) and kNN-CNI(C4.5) each achieve about 2–4% higher average classification accuracies than kNN on the original incomplete datasets, with 1-sided paired t-test $p < 0.0006$

Table 2: Average Classification accuracy over the 12 datasets for instance-based algorithms using different nominal imputers as preprocessors (from top to bottom, bolded are best values) (a) kNN using (1-distance) weighting, (b) kNN using $1/distance$ weighting, (c) kNN without distance weighting.

Missing Ratio(%)	Original kNN	kNN-MCI	kNN-CNI(C4.5)	kNN-CNI(SVM)	biggest-lift (%)
10	83.02	83.94	84.27	**84.55**	1.85
20	80.72	81.35	**83.37**	83.06	3.28
30	78.48	80.01	80.83	**81.36**	**3.67**
40	76.14	78.11	78.57	**78.92**	3.65
50	74.14	75.54	**76.00**	75.86	2.51
average	78.5	79.79	80.61	**80.75**	2.99

Missing Ratio(%)	Original kNN	kNN-MCI	kNN-CNI(C4.5)	kNN-CNI(SVM)	biggest-lift (%)
10	82.44	83.99	84.36	**84.55**	2.57
20	81.21	82.06	**83.32**	82.92	2.6
30	79.2	79.97	80.82	**81.37**	2.74
40	76.6	78.03	78.57	**78.93**	**3.04**
50	73.89	75.57	**76.02**	75.9	2.88
average	78.67	79.93	80.62	**80.73**	2.76

Missing Ratio(%)	Original kNN	kNN-MCI	kNN-CNI(C4.5)	kNN-CNI(SVM)	biggest-lift (%)
10	82.67	83.72	83.97	**84.78**	2.56
20	80.57	81.94	**83.13**	82.38	3.17
30	78.34	79.61	80.77	**81.39**	**3.9**
40	76.17	77.88	78.64	**78.90**	3.58
50	73.73	75.34	**75.72**	75.7	2.7
average	78.3	79.7	80.44	**80.63**	3.18

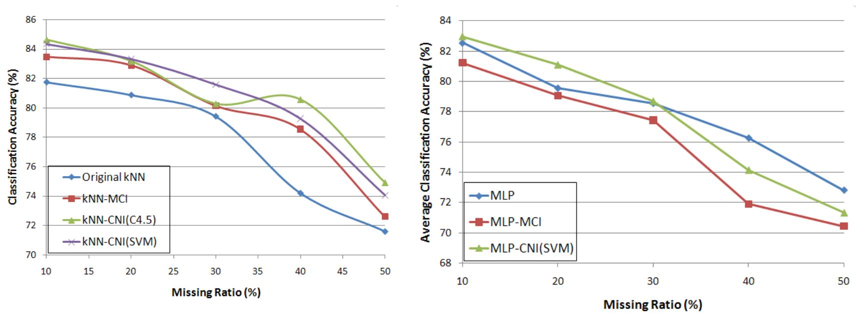

Fig. 4: (from left to right) (a) Case study on the dataset "Australian": using nominal imputers to preprocess incomplete training data helps improve the classification performance; and a better imputer (such as CNI(SVM)) will give a bigger lift of the classification accuracy. (b) The average classification accuracy of MLP, and MLP using MCI and CNI(SVM) as preprocessor for incomplete training data over all 12 datasets.

and $p < 0.001$ respectively. KNN using MCI as the preprocessor slightly outperforms the original kNN by 1–2% on average, with $p < 0.01$.

3.3 The Impact of Nominal Imputers on Other Machine Learned Classifiers

We have shown that using high-quality imputation techniques to preprocess incomplete data will increase the classification performance for instance-based algorithms on nominal data. We now further investigate whether nominal imputers can help other classifiers perform better on incomplete nominal data, especially those with better missing data handling schemes – ie, where imputation may not be as critical.

We work on three machine learning algorithms, naïve Bayes [6], C4.5 [8], and MLP neural network [5], on the 12×5 datasets. We use one baseline imputer, MCI, and one top performer of our CNI imputers, CNI(SVM), as the nominal imputers for the training data before learning the classifiers. We compare with the classifiers that do not use imputation before training. Table 3 summarizes the results, and Figure 4(b) depicts the average accuracy for MLP, MLP-MCI, and MLP-CNI(SVM) on datasets of different missing ratios.

As Table 3 and Figure 4(b) indicate, accurate nominal imputers such as CNI(SVM) can help improve classification accuracy when the incomplete data has low missing ratios (e.g., at or below 20%), while using an inaccurate imputer such as MCI may lead to worse classification performance than the classifier that does not use any imputation for training data. When the missing ratio goes higher (i.e., missing ratio higher than 30%), neither CNI imputers nor MCI can help the learner to improve its classification accuracy. This is partially because many classifiers, such as NB, have more effective ways to deal with incomplete data than using imputation for highly sparse data. When the missing ratio is

Table 3: The average classification accuracies of the machine learned classifiers NB, C4.5, and MLP with and without using nominal imputers MCI and CNI(SVM)

	10.00%	20.00%	30.00%	40.00%	50.00%	average
NB	83.31	83.03	**81.85**	**80.65**	**77.82**	81.33
NB-MCI	82.73	81.87	81.07	79.07	75.43	80.03
NB-CNI(SVM)	**83.35**	**83.14**	81.62	79.15	76.69	80.79
C4.5	83.27	80.51	78.85	75.96	71.25	77.97
C4.5-MCI	83.31	**81.28**	**79.15**	**76.08**	**74.09**	78.78
C4.5-CNI(SVM)	**84.40**	81.11	77.56	75.34	71.77	78.04
MLP	82.55	79.56	78.56	**76.25**	**72.80**	77.95
MLP MCI	81.22	79.07	77.44	71.91	70.44	76.01
MLP -CNI(SVM)	**82.98**	**81.10**	**78.71**	74.17	71.34	77.66

high, the nominal imputers will become less accurate (see Section 3.1), and the benefit of using imputation will be offset by its inaccuracy.

Why are accurate nominal imputers effective for instance-based learning algorithms on both high-missing-ratio data and low-missing-ratio data? We suspect that this is partially due to the poor way that instance-based algorithms handle missing data, as using the maximized distance calculation can lead to poor classification accuracy, while using CNI imputation to preprocess incomplete nominal data before learning the base classifier will perform better.

As many real-world dataset are missing under 20% of the values, there is practical significance for our finding that an accurate nominal imputer (such as our proposed CNI imputation) can improve the classification performance of many machine learners on such incomplete datasets, even over learners that include effective built-in schemes for handling missing data.

4 Conclusions

Incomplete nominal data often make it difficult for learning algorithms to produce effective classifiers. Simple schemes of imputing the missing values, such as using the most common value for nominal missing data, are often not very effective. In this paper, we explore methods for improving classification performance for machine learning algorithms on incomplete nominal data. We first propose an easy-to-implement and effective nominal imputation algorithm: classifier-based nominal imputation (CNI), which fills in the missing values of attribute f_i by the values produced by a learned classifier that takes the other values in that instance, where this classifier is learned using as the training data the instances that contain the observed values of f_i. Our empirical results show that, of the 10 classification algorithms for imputation we investigated, the support vector machine (SVM) and C4.5 decision tree perform the best as CNI imputation learners. Our CNI imputers have significantly higher estimation accuracy than the commonly used nominal imputation techniques, kNN imputation (kNNI) and most common imputation (MCI). Applying CNI imputers such as CNI(SVM) and CNI(C4.5) as the preprocessors for incomplete nominal training data can impressively improve the classification performance of instance-based learning algorithms on incomplete datasets over the entire range of missing ratios that we investigated. CNI imputers are also found helpful in improving the classification performance of machine learners that have effective missing data schemes, such as NB and C4.5, when the datasets have missing ratios at or below 20%.

In our future work, we plan to improve our CNI imputation algorithm by using the previously imputed values for the later iterations of imputations, in the order of imputations based on how dense or informative the attributes are. We also plan to explore ways to improve the instance-based learning algorithm using nominal imputations for other data types, such as mixed data (with both numeric and nominal data) and ordinal data.

Acknowledgement

This work was conducted when X. Su was with Varolii Corporation. R. Greiner was supported by the Alberta Ingenuity Centre for Machine Learning (AICML) and Canada's Natural Science and Engineering Research Council (NSERC). This work was supported by the Data Mining and Machine Learning Laboratory at Florida Atlantic University. We also thank Randall Wald for his help.

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning 6, 37–66 (1991), http://dx.doi.org/10.1007/BF00153759, 10.1007/BF00153759
2. Batista, G., Monard, M.: An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence 17(5), 519–533 (2003)
3. Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml
4. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11(1), 10–18 (2009)
5. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edn. (1998)
6. John, G., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proceedings of the eleventh conference on uncertainty in artificial intelligence. vol. 1, pp. 338–345. Citeseer (1995)
7. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization, pp. 185–208. MIT Press, Cambridge, MA, USA (1999), http://portal.acm.org/citation.cfm?id=299094.299105
8. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
9. Su, X., Khoshgoftaar, T., Greiner, R.: Using imputation techniques to help learn accurate classifiers. In: Tools with Artificial Intelligence, 2008. ICTAI '08. 20th IEEE International Conference on. vol. 1, pp. 437 –444 (Nov 2008)