VipBoost: a More Accurate Boosting Algorithm

Xiaoyuan Su
Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
xsu@fau.edu

Taghi M. Khoshgoftaar
Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
taghi@cse.fau.edu

Russell Greiner
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
greiner@cs.ualberta.ca

Abstract

Boosting is a well-known method for improving the accuracy of many learning algorithms. In this paper, we propose a novel boosting algorithm, VipBoost (voting on boosting classifications from imputed learning sets), which first generates multiple incomplete datasets from the original dataset by randomly removing a small percentage of observed attribute values, then uses an imputer to fill in the missing values. It then applies AdaBoost (using some base learner) to produce classifiers trained on each of the imputed learning sets, to produce multiple classifiers. The subsequent prediction on a new test case is the most frequent classification from these classifiers. Our empirical results show that VipBoost produces very effective classifiers that significantly improve accuracy for unstable base learners and some stable learners, especially when the initial dataset is incomplete.

1. Introduction

Ensemble techniques are popular as they can generally produce fairly accurate classifiers. Two well-known ensemble techniques are *boosting* [1] and *bagging* [2], which learn diverse classifiers, then (at performance time) combine their responses for each test instance.

In this work, we propose to inject diversities into the learning set by randomly removing observed values multiple times, to produce n different incomplete learning sets. We then use an imputation technique to fill in the missing values to produce n different *complete* training sets, then apply AdaBoost [1] (using some base learner) on each of the imputed training sets, to produce n different classifiers. For predicting new test cases, each of these classifiers returns a classification label for an instance; our system returns the most frequent label.

While removing some attribute values may cause the resulting learned classifier to make more mistakes, existing research into ensemble methods has shown that many ensemble methods work best when the base classifiers are diverse – i.e., when their errors are fairly independent [3]. Moreover, our preliminary experiments show that the accuracy of the classifier trained from an incomplete dataset, produced by removing a small number of attribute values, is often close to, or occasionally better than, the classifiers trained from the original dataset.

For incomplete data, the classifiers learned with the help of an imputation technique is often more accurate than one obtained by just applying some standard learner directly to the original data [4]. By boosting a base learner on these imputed learning sets, we may get the further accuracy improvement from the boosting method. Therefore, by voting on boosting classifications from different imputed learning sets originated from the same underlying dataset, we expect that our *VipBoost* (voting on boosting classifications from imputed learning sets) predictors can outperform conventional machine learners, bagging predictors, and boosting predictors (i.e., AdaBoost [1]), especially for incomplete data.

Here, we randomly removed m% of the attribute values over the entire training data (here, m=5). Note this is the "Missing Completely at Random" (MCAR) mechanism, as these removals are independent of the actual value of this feature, and of any other feature.

We evaluate the predictive performance of VipBoost using the following 10 base learners (from WEKA [5]), including both unstable learners: decision tree (C4.5), decision table (dTable), logistic regression (LR), neural networks (NN), one rule (OneR), decision list (PART), support vector machine (SVM); and stable learners: naïve Bayes (NB), k-nearest-neighbor (kNN), and random forest (RF). We considered three imputation techniques: expectation maximization (EM) [6][7], Bayesian multiple imputation (BMI) [8], and mean imputation (MEI, the baseline imputation technique).

Section 2 is the framework of our VipBoost algorithm. Experimental design and results are in Section 3, and Section 4 presents our conclusions.

2. Framework

Our VipBoost algorithm first produces *n* incomplete datasets by removing *m*% of the observed values from the initial dataset, completely at random, *n* times. (If the initial dataset was missing *k*% of its values, including *k=0* for complete data, then each new dataset is missing *(k+m)*% of the values.) VipBoost then imputes the missing values using some imputation techniques *Imp*, such as EM and BMI, to fill in the missing values; it then applies *AdaBoost* (with some base learner *L*) [1] to each of the imputed learning sets to produce *n* different classifiers. It then returns a single classifier that runs each of these classifiers, and returns the most frequent classification. See Figure 1.

Algorithm: VipBoost(L, Imp D, n, m)

Input: *L*: base learner (e.g., kNN, SVM, …)
Imp: imputation method (e.g., EM, BMI, …)
D: a labeled training dataset with *k*% missing values (k≥0)
n: number of generated learning sets with injected missing values
m%: the injected missing ratio
Output: a classifier

Begin
For i=1, 2,…, *n*
{
a. Remove *m*% observed attribute values in *D* completely at random to generate the *i*-th learning set D_i
b. $D_i' = Imp[D_i]$ -- an imputed dataset;
c. $c_i = AdaBoost[L](D_i')$
}
Return the classifier $c^*(\mathbf{x}) = \text{plurality}(c_i(\mathbf{x}))$
% for any x, $c^*(x)$ return the most frequent class label.
End

Figure 1. The framework of VipBoost algorithm

We investigated *VipBoost[L, Imp D, n, m]* for 10 machine learning algorithms *L*, each of which has its own method of dealing with missing values. Some just ignore missing values during the learning and classifying processes, e.g. Naïve Bayes (NB) [9], decision table (dTable) [10], decision list (PART) [11], and decision tree (C4.5) [12]. Some replace the missing values with the mean or median value for numeric attributes or the most frequent value for nominal attributes, e.g., Logistic regression (LR) [13] and Random Forest (RF) [14]. Some – including the one-rule classifier (OneR) [15], and the sequential minimal optimization (SMO) [16] (*WEKA's* implementation of support vector machine, SVM) – treat missing values as a legitimate value ("*missing*"). K-nearest-neighbor (kNN) counts missing values into its distance measure (Euclidean distance for continuous attributes and Hamming distance for discrete ones). When the two instances each omit the values of the same attribute, kNN sets the distance wrt that attribute to 0, but when only one has a missing value, kNN sets the value to a maximal distance [5]. Neural network (NN) replaces the missing values using an interval (eg., a unit interval [0,1] that includes all the possible values of that attribute) [17].

As these simple approaches to dealing with missing values may not significantly improve classification performance, VipBoost uses a different imputation technique *Imp* before handing the completed data to *AdaBoost [L]*. We investigated using the state-of-the-art imputation techniques EM and BMI, as well as the baseline mean imputation, to impute the incomplete learning sets.

Mean Imputation

In general, let $Y_{u,i}$ be the observed value of the *u*-th instance on the *i*-th attribute. Mean imputation (MEI) fills in a missing value $Y_{v,i}$ with the mean of the observed values on each attribute

$$\hat{Y}_{v,i}^{MEI} = \frac{1}{|U(i)|}\sum_{u\in U(i)} Y_{u,i} \qquad (1)$$

where *U(i)* is a set of indices of instances that have observed values of attribute *i*. MEI rounds a mean estimate to the nearest integer for discrete values. MEI is used by many machine learning algorithms already (e.g., LR), as it is the simplest imputation technique. However, it distorts the shape of distributions by creating a spiked distribution at the mean in frequency distributions, and it also reduces (under-estimates) the variance of the predictions, which typically leads to wrong inferences.

Expectation Maximization Imputation

In general, expectation-maximization (EM) seeks maximum likelihood estimates of parameters in probabilistic models in the presence of latent variables [6][7]. EM iterates between performing an expectation *E*-step, which calculates an expected value of the complete data likelihood, given the observed data and the current parameters; and a maximization *M*-step, which computes values of the parameters that maximize the expected likelihood over the data, including those estimated in the E-step. The parameters found on the *M*-step are then used to

begin another *E*-step, and the process is repeated until it converges to a stationary point. EM imputation requires specifying a joint probability distribution for the attribute value to be imputed and other attribute values.

Our implementation of the EM imputation algorithm for multivariate Gaussian data (parameterized by the mean and the covariance matrix) uses ridge regression [7]. It first gives an initial guess of these parameters. In each following iteration, EM then iterates: (1) fill each missing value with its conditional expectation value given the observed values in that instance using the estimated mean and covariance matrix; (2) re-estimate the mean and the covariance matrix, using the instance mean of the completed dataset and the covariance matrix as the sum of the instance covariance matrix of the completed dataset and the contributions from the conditional covariance matrix of the imputation errors. EM repeats these steps until the imputed values and the estimates of the mean and covariance do not change [18].

We round EM imputed values to the nearest integers for integer attributes. We also find the observed value range [*min*, *max*] for each attribute, and replace imputed values <*min* with *min*, and those >*max* with *max* for missing values. We also apply this post-processing procedure to the BMI imputer described below.

Bayesian Multiple Imputation

Standard *single imputation* produces a single imputed dataset, where each missing value is replaced with a single value. While this approach can be applied to virtually every dataset, single imputation does not account for the uncertainty about the predictions of the imputed values; this can lead to statistically invalid inferences [8]. By contrast, *multiple imputation* (MI) produces many different imputed datasets. In many situations, MI approaches have proven to be highly effective even for small values of *m* – say 3 to 10 [8].

BMI follows a Bayesian framework: it specifies a parametric model for the complete data, with a given *a prior* distribution over the unknown model parameters θ, then simulates *m* independent draws from the conditional distribution of the missing data given the observed data. In non-trivial applications, special computational processes, such as Markov chain Monte Carlo (*MCMC*), must be applied to perform Bayesian multiple imputation [8]. While BMI assumes a multivariate normal distribution when generating the imputations for missing values, it is robust to non-normally distributed data [19].

Let $P(Y_{com}|\theta)$ model the complete data, based on the parameter θ (the mean and covariance matrix that parameterizes a normal distribution). If $Y=(Y_{obs}, Y_{miss})$ follows a parametric model $P(Y|\theta)$ with θ having the prior distribution $P(\theta)$, then the posterior predictive distribution for Y_{miss} is

$$P(Y_{miss} \mid Y_{obs}) = \int P(Y_{miss} \mid Y_{obs}, \theta) P(\theta \mid Y_{obs}) d\theta \quad (2)$$

Equation 2 suggests that *BMI* can be drawn by repeating the following process for $j=1, \ldots, n$:

(1) generate missing values $Y_{miss}^{(j+1)}$ from $P(Y_{miss}|Y_{obs}, \theta^{(j)})$;

(2) draw parameters $\theta^{(j+1)}$ from $P(\theta|Y_{obs}, Y_{miss}^{(j+1)})$.

These two steps are repeated to generate the Markov chain $\{Y_{miss}^{(1)}, \theta^{(1)}, Y_{miss}^{(2)}, \theta^{(2)}, \ldots, Y_{miss}^{(j)}, \theta^{(j)}, \ldots\}$ (here $Y_{miss}^{(j+1)}$ depends on $\theta^{(j)}$, and $\theta^{(j)}$ depends on $Y_{miss}^{(j)}$). This iterative process continues until the distribution $P(Y_{miss}, \theta \mid Y_{obs})$ is stabilized. This produces the values Y_1^C; the then run the same procedure many times, to produce *m* completed datasets $\{Y_i^C\}$. We then take the average as the final imputed dataset,

$$Y^C = \frac{1}{m} \sum_{i=1}^{m} Y_i^C \quad (3)$$

3. Experimental Design and Results

We worked on 10 datasets *D* with numeric or ordinal attributes from the *UCI* machine learning repository [20], half of which have binary classes, and five others have multiple classes (see Table 1). We applied our VipBoost framework to the 10 commonly used machine learners *L* listed in Section 1. For each dataset, we generated n=9 incomplete training sets and used the default value (I=10) as the maximum number of AdaBoost iterations in the boosting stage of VipBoost for each dataset. We use m=5% as *injected missing ratio* for each dataset. We use k=5 for kNN, and default parameters of WEKA for all other machine learned classifier.

Table 1. Description of the datasets used

Datasets	Train #	Test #	attri #	class #
australian	460	230	14	2
breast-wisc	466	233	10	2
diabetes	512	256	8	2
heart	180	90	13	2
letter	15000	5000	16	26
pima	512	256	8	2
satimage	4290	2145	36	6
segment	1540	770	19	7
vehicle	564	282	18	4
waveform	300	4700	21	3

We compare VipBoost predictors with original machine learned classifiers, bagging predictors, AdaBoost on each of the learners, and used the imputers *Imp* in {BMI, EM, MEI}. We first investigate the performance of the various learners *L* when the initial training data are complete, in terms of classification accuracy over the 10 datasets (see Table 2, and Figure 2). We then considered initially incomplete data, missing 30% of the values (generated by randomly removing observed attribute values from the complete datasets, see results in Table 3 and Figure 3). We also investigate the classification performance on a single dataset "waveform" with missing ratios from 0% (complete data) to 80%; see Table 4 and Figure 4.

We considered the average performance over all 10 learners *L* and 10 datasets *D*. On complete data, among VipBoost with the three imputers *Imp*, we found that VipBoost-BMI was the best , as it on average had 2.5%, 1.6%, and 1.4% higher average classification accuracy than the original classifiers, AdaBoost, and bagging predictors (overall average) respectively, with 1-sided t-test p<1.4E-5, p<2.0E-9, and p<8.0E-5 respectively (see Table 2 and Figure 2 for average classification accuracy over 10 datasets; stable learners are highlighted). At the same time, VipBoost does not improve the accuracy on stable algorithms kNN, naïve Bayes, and random forest, which is similar to bagging and boosting here.

On incomplete datasets, VipBoost predictors have much bigger advantages over original classifiers, bagging predictors, and AdaBoost than on the complete data. For the datasets missing 30% of the values (see Table 3 and Figure 3 with stable learners highlighted), VipBoost-BMI, VipBoost-EM, and VipBoost-MEI have 11.0%, 10.5%, and 5.8% higher average classification accuracy than the original learners respectively. AdaBoost and bagging predict-

Table 2. Average classification accuracy over 10 complete UCI datasets

	classi -fiers	Ada Boost	Bagg -ing	Vip Boost -MEI	Vip Boost -EM	Vip Boost -BMI
OneR	62.95	65.56	64.28	66.64	67.53	**67.98**
NB	**75.98**	75.5	75.7	75.91	75.61	75.03
dTable	78.18	81.61	80.65	83.64	84.15	**84.03**
C4.5	82.27	83.8	84.79	86.08	85.92	**86.49**
kNN	**83.4**	82.11	82.27	82.39	82.52	82.64
PART	82.51	84.22	84.59	85.96	86.46	**86.61**
SVM	83.32	83.76	**84.42**	83.49	83.81	83.94
LR	84.28	83.92	84.22	84.22	84.1	**84.5**
NN	83.71	84.43	85.08	85.59	86.2	**85.84**
RF	**86.31**	84.72	85.33	85.65	86.23	85.79
Ave	80.29	80.96	81.13	81.95	82.25	**82.28**

Table 3. Average classification accuracy over 10 UCI datasets with 30% missing ratio

	classi -fiers	Ada Boost	Bagg -ing	Vip Boost -MEI	Vip Boost -EM	Vip Boost -BMI
OneR	56.33	58.27	59.09	59.95	68.03	**68.06**
NB	73.35	73.02	**73.58**	71	72.41	72.95
dTable	66.35	71.02	65.36	75.7	78.32	**78.67**
C4.5	73.93	77.07	77.63	79.25	80.67	**81.64**
kNN	57.61	48.96	48.38	71.76	78.15	**79.36**
PART	75.12	78.13	79.88	79.92	81.12	**81.7**
SVM	74.68	75.04	74.82	75.49	78.78	**78.83**
LR	75.03	75.15	74.36	74.49	**78.19**	78.17
NN	70.72	72.01	74.77	76.4	**80.01**	79.92
RF	80.18	79.99	80.45	79.81	81.32	**81.43**
Ave	70.33	70.86	70.83	74.38	77.7	78.07

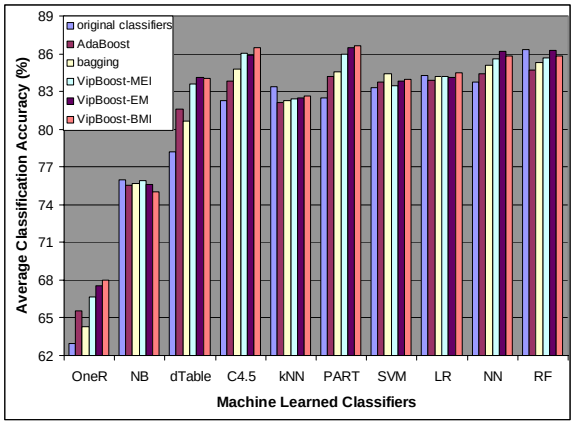

Figure 2. Average classification accuracy over 10 complete UCI datasets

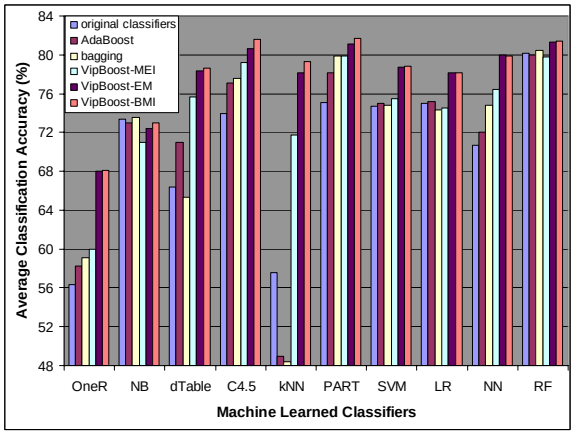

Figure 3. Average classification accuracy over 10 UCI datasets with 30% missing ratio

ors have limited improvement here (with 0.71% and 0.75% respectively). VipBoost-BMI and VipBoost-EM on average outperform AdaBoost by 10.2% and 9.7% respectively.

Note that on incomplete data, VipBoost predictors overcome the property that most ensemble classifiers (e.g., bagging and boosting) can not improve stable processes. VipBoost-BMI, VipBoost-EM, and VipBoost-MEI greatly improve the classification performance of the original kNN, a stable learner, on incomplete data, with 37.8%, 35.7%, and 24.4% higher average classification accuracy respectively, partly due to the crude missing value handling by kNN (see Section 2). Bagging and AdaBoost have decreased classification performance for kNN here. VipBoost predictors also improve another stable learner, random forest. VipBoost-BMI and VipBoost-EM improve the original random forest on average by 1.6% and 1.4%. However, they do not improve naïve Bayes, which just ignores missing values in both learning and classification stages.

Table 4 and Figure 4 show the average classification accuracy of our VipBoost predictors, AdaBoost, bagging predictors, and original machine learned classifiers on the dataset "waveform", which has different MCAR missing ratios from 0% (complete data) to 80%. VipBoost-BMI and VipBoost-EM perform the best for datasets with all applicable missing ratios. Note even VipBoost-MEI, the VipBoost predictor using the simplest imputation technique MEI, outperforms the well-known ensemble classifiers bagging predictors and AdaBoost. Because of eigenvector calculation exception, VipBoost-EM does not work for datasets with higher missing ratio (i.e., missing ratios > 50%).

Table 4. Average classification accuracy over 10 classifiers on the dataset "waveform" with missing ratios 0%~80%

Miss Ratio %	classi -fiers	Ada Boost	Bagg -ing	Vip Boost -MEI	Vip Boost -EM	Vip Boost -BMI
0	75.64	78.91	77.9	81.21	**81.49**	81.35
10	72.86	76.16	74.98	79.27	80.56	**80.9**
20	71.21	73.76	73.44	78.08	80.14	**80.72**
30	68.3	70.18	69.67	74.77	**79.68**	79.58
40	66.67	68.16	68.47	71.85	77.71	**78.22**
50	63.91	65.9	64.63	68.46	76.26	**76.32**
60	58.53	59.81	59.87	63.72	NA	**67.03**
70	56.82	57.11	56.82	60.42	NA	**63.03**
80	50.67	50.61	51.16	55.34	NA	**57.82**

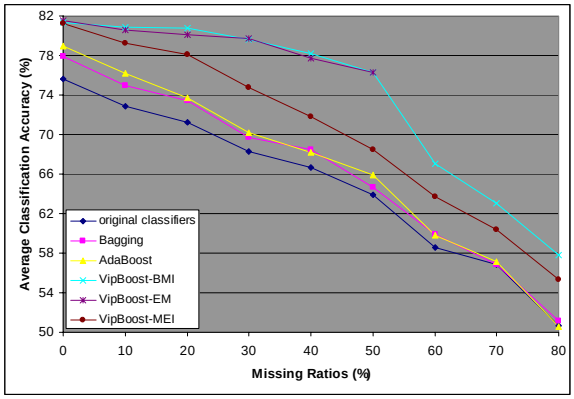

Figure 4. Average classification accuracy over 10 classifiers on the dataset "waveform" with missing ratios 0%~80%

The effectiveness of VipBoost depends on the imputation technique *Imp*, the base learner *L*, the missing data rate, and also the *injected missing ratio m*. Our empirical study shows that an *m* between 3%~8% is effective. We use *m*=5% as the default *injected missing ratio* in this work.

4. Conclusions

We propose a novel ensemble algorithm: VipBoost (voting on boosting classifications from imputed learning sets). VipBoost injects diversity to the baseline learning set by randomly removing observed attribute values multiple times and then imputing each of the resulting datasets. It makes the final classification by voting on boosting classifications, learned from the various imputed training sets. Our experimental results show that VipBoost predictors significantly improve the classification performance of conventional unstable learners, and the well-known AdaBoost and bagging predictors, especially for incomplete data. VipBoost can also significantly improve the classification performance for some stable learners, especially kNN, which traditional ensemble classifiers such as AdaBoost and bagging predictors fail to improve.

References

[1] Freund, Y., and Schapire, R.E., A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, 55(1), pp. 119-139, 1997.

[2] Breiman, L., Bagging Predictors, *Machine Learning*, 24(2), 1996.

[3] Kuncheva, L., and Whitaker, C. Measures of Diversity in Classifier Ensembles and their Relationship with the Ensemble Accuracy, Machine Learning, 51(2), 2003.

[4] Su, X., Khoshgoftaar, T.M., and Greiner, R., Using Imputa-tion Techniques to Help Learn Accurate Classifiers, the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 1, pp. 437-444. 2008.

[5] Witten, I.H., and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann, 2005.

[6] Dempster, A.P., Laird, N.M., and Rubin, D.B., Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society*, B 39, pp. 1-38, 1977.

[7] Schneider, T., Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values, *Journal of Climate*, 14, pp. 853–871, 2001.

[8] Rubin, D.B., Multiple Imputation for Nonresponse in Surveys. *J. Wiley & Sons*, New York, 1987.

[9] John, G.H. and Langley, P. Estimating Continuous Distributions in Bayesian Classifiers, *UAI*, pp.338-345, 1995.

[10] Kohavi, R. The Power of Decision Tables, *ECML*, 1995.

[11] Frank, E. and Witten, I.H. Generating Accurate Rule Sets Without Global Optimization. In Shavlik, J., ed., Machine Learning: Proceedings of the Fifteenth International Conference, Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[12] Quinlan, J.R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993

[13] Cessie, S. le. and Houwelingen, J.C. van, Ridge Estimators in Logistic Regression, *Applied Statistics*, 41(1), pp. 191-201, 1992.

[14] Breiman, L. Random Forests. *Machine Learning*, 45(1), pp. 5-32, 2001.

[15] Holte, R.C. Very simple classification rules perform well on most commonly used datasets, *Machine Learning*, 11, pp. 63-91, 1993.

[16] Platt, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization, *Advances in Kernel Methods Support Vector Learning*, 1998.

[17] Ishibuchi, H., Miyazaki, A., Kwon, K., and Tanaka H. Learning from Incomplete Training Data with Missing Values and Medical Application, *IJCNN*, pp. 1871-1874, 1993.

[18] Little, R.J.A. and Rubin, D.B. Statistical Analysis with Missing Data. *Series in Probability and Mathematical Statistics*, Wiley, pp. 278, 1987.

[19] Schafer, J.L., Analysis of Incomplete Multivariate Data, *New York: Chapman and Hall*, 1997.

[20] Blake, C., and Merz, C. *UCI Repository of Machine Learning Databases*, 2000. http://www.ics.uci.edu/~mlearn/MLRepository.htm