DualBoost : Handling Missing Values with Feature Weights and Weak Classifiers that Abstain

Weihong Wang, Jie Xu, Yang Wang, Chen Cai and Fang Chen UTS and Data61 CSIRO

INTRODUCTION

In real world datasets, missing values are a common issue. They are introduced due to various reasons, including data corruption, data unavailability or erroneous manual inputs. Handling missing values is one of the most key aspects in data mining, as inappropriately handled missing values can cause biased estimates in a statistical analysis or seriously impact the performance of predictive models, leading to invalid conclusions.

AIM

The easiest method to handle missing values is to ignore data samples with missing values. However, this method loses all the information in the discarded data samples, and is not applicable when most of data samples contain missing values. Another method is to impute missing values. For example, replacing a missing value with the median (for numerical values) or the mode (for categorical values) of all data samples. The method prevents information loss but brings artificial variance and bias to the data. The third method is to apply machine learning models directly to data samples

METHODS

This study proposes a unified Boosting framework called DualBoost that consolidates model construction and missing value handling. At each Boosting iteration, weights are assigned to both the samples and features. The sample weights make difficult samples become the learning focus, while the feature weights enable critical features to be compensated by less critical features when they are unavailable. A weak classifier that abstains is learned on a data subset determined by the feature weights. Experimental results demonstrate the efficacy and robustness of the proposed method over existing Boosting algorithms.

Algorithm 1 DualBoost

Input:

- $(x, y)^M$: M training samples $(x_1, y_1), \cdots, (x_m, y_m), \cdots, (x_M, y_M)$ with N features where $y_m = -1, 1$ for negative samples and positive samples respectively
- T: number of iterations
- S: number of iterations for nullifying features

Output: a Boosting classifier $F(x)$

1: Initialize sample weights: $sw_m \leftarrow \frac{1}{M}, m = 1...M$.
2: Initialize feature weights: $fw_n \leftarrow \frac{1}{N}, n = 1...N$.
3: Train a classifier f_0 from $(x, y)^M$ and obtain its error ε_0.
4: **for** $t = 1 \rightarrow T$ **do**
5: Normalize the sample weights: $sw_m \leftarrow \frac{sw_m}{\sum_{j=1}^{M} sw_j}$.
6: Calculate feature distribution $p \leftarrow \frac{fw}{\sum_{i=1}^{N} fw_i}$
7: $(dx, y)^M$, $\varepsilon_{fm} \leftarrow$ DE-EMPHASISE$((x, y)^M, p, t, T, S, \varepsilon_{t-1}, f_{t-1})$
8: Train a pool of weak classifier $\{f_j(x)\}$ on $(dx, y)^M$. With the current sample weights sw, calculate the total weight of the training examples on which it abstains as W_a, that it classifies correctly as W_c and that it misclassifies as W_m.
9: Find the classifier f_t which minimizes $Z \leftarrow W_a + 2\sqrt{W_c W_m}$. Check that $W_c > W_m$.
10: The corresponding error of f_t: $\varepsilon_t \leftarrow W_m$.
11: Calculate $\alpha_t \leftarrow \frac{1}{2} \ln(W_c / W_m)$.
12: Update sample weights: $sw_m \leftarrow sw_m \exp(-\alpha_t y_m f_t(x_m))$.
13: $I_t \leftarrow$ IMPORTANCE$((x, y)^M, f_t)$
14: Update feature weights: $fw_n \leftarrow fw_n + I_{t,n} * (\varepsilon_{fm} - \varepsilon_t)$
15: **end for**
16: The final strong classifier is

$$F(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t f_t(x) \geq 0; \\ -1 & \text{otherwise.} \end{cases}$$

Algorithm 2 DE-EMPHASISE

Input:

- $(x, y)^M$: M training samples with N features
- p : a distribution over N features
- t: current iteration number
- T: total number of Boosting iterations
- S: total number of iterations for nullifying features
- ε: previous hypothesis error
- f: previous hypothesis

Output:

- $(dx, y)^M$: $(x, y)^M$ with K features nullified using p
- ε_{fm}: the classification error resulted by nullifying N features from $(x, y)^M$

1: $\varepsilon_{fm} \leftarrow$ NULLIFY-FEATURES$((x, y)^M, p, N, S, f)$
2: $\varepsilon_m \leftarrow \epsilon$
3: $K \leftarrow 0$
4: **while** $\varepsilon_m < \frac{(T-t)*\epsilon + t*\epsilon_{fm}}{T}$ and $\frac{K}{N} < 0.90$ and $\frac{K}{N} > 0.15$ **do**
5: $K \leftarrow K + 1$
6: $\epsilon_m \leftarrow$ NULLIFY-FEATURES$((x, y)^M, p, K, S, f)$
7: **end while**
8: $(dx, y)^M \leftarrow (x, y)^M$ with K features nullified using p

RESULTS

We present the results of the proposed DualBoost algorithm on benchmark data sets, and compare them with the results of existing methods, AdaBoost and AdaBoost with weak classifers that abstain (AdaBoost-ABS) to demonstrate DualBoost's improvement of performance and robustness. We use two real-world datasets, namely Banknote authentication and Wisconsin Breast Cancer
datasets, from the UCI repository.

We use decision trees that abstain to deal with missing features values in the data set.

❑ Without Data Imputation

In this experiment, we compare DualBoost and AdaBoost-ABS algorithms on the Banknote authentication data set. Both algorithms can be directly applied to data with missing values without imputation. The results are shown in Figure 1.

❑ With Data Imputation

In this experiment, we compare DualBoost, AdaBoost-ABS and AdaBoost algorithms on the Breast Cancer authentication data set. Both DualBoost and AdaBoost-ABS can be directly applied to data with missing values. As AdaBoost cannot be applied to data with missing values, we apply data imputation to both the training and test data for AdaBoost. The results are shown in Figure 2.

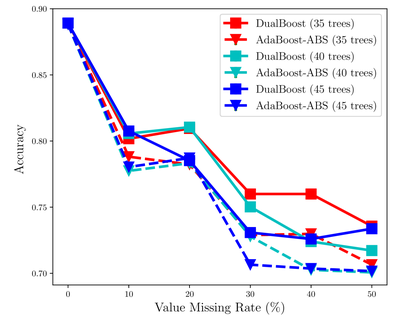

Figure 1: The experiment results on the Banknote authentication data set.

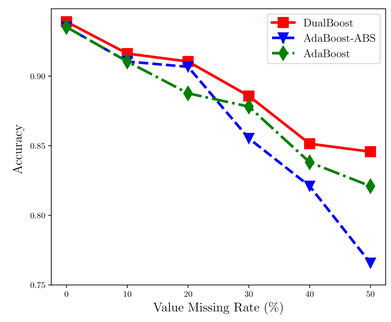

Figure 2: The experiment results on the Breast Cancer data set.

CONCLUSIONS

In this paper we extend the Boosting framework to tackle missing values. Specifically weights are assigned to samples and features. The sample weights guide the learning on difficult samples while the feature weights enables the compensation of critical features with less critical features. Furthermore, we employ weak classifiers that abstain on data subsets that are determined by feature weights , which leads to robust missing value handling.