Learning Interpretable Classification Rules Using Binarization

Yuka Yoneda [1] **Mahito Sugiyama** [2 3] **Takashi Washio** [1]

Abstract

We propose to learn classification rules from continuous data with class labels using a two-step procedure: We first binarize data points, and second perform feature selection on the binarized data to obtain minimum necessary binary features that are required to classify given data. Each binary feature represents an interval on the original feature space, hence the user can directly interpret classification rules as intersections of intervals. Although such *interpretability* is indispensable in machine learning applications from biology to economics, we cannot often interpret classification rules obtained by recently emerging highly accurate machine learning methods such as deep learning. We experimentally demonstrate that our method can learn simpler classification rules than a decision tree classifier while keeping reasonable accuracy.

1. Introduction

Machine learning methods tend to derive complex classification rules in order to achieve high classification accuracy when we perform supervised classification on continuous data. For example, a support vector machine (SVM) classifies data points by a hyperplane on a high-dimensional implicit feature space, thereby it is difficult to understand classification rules. For another example, recent deep learning approaches (LeCun et al., 2015) can achieve high classification accuracy, but the way how it classifies data points is hard to understand. In contrast, decision tree based supervised learning methods (Han et al., 2011, Chapter 8.2) such as CART (Breiman et al., 1984), ID3 (Quinlan, 1986), and C4.5 (Quinlan, 1993) learn classification rules for continuous data by dividing the entire feature space into rectangular areas. Although we can easily understand classification rules obtained by such decision tree based methods, when we classify high-dimensional continuous multivariate data, they often fail to achieve high accuracy or obtain concise classification rules. Thus, there is a need to develop a machine learning method which achieves sufficiently high accuracy while obtaining interpretable simple rules for continuous data.

Figure 1. Concept diagram of proposed method.

In order to solve the problem, we propose to combine binarization of continuous data and feature selection on the binarized data. First, we discretize continuous data and convert them into binary data. Next, we apply Super-CWC algorithm (Shin et al., 2015) to the binarized data, which is an efficient feature selection method for binary data and enables us to obtain a minimal binarized feature set to distinguish data points according to given class labels. Finally, we obtain a combination of binarized features, which corresponds to the intersection of rectangular regions on the original feature space and can be used to classify unlabeled data. Figure 1 shows the conceptual diagram of our proposed method.

The remainder of this paper is organized as follows. Section 2 introduces our proposed method; Section 2.1 explains binarization and Section 2.2 introduces the Super-CWC algorithm. Section 3 empirically examines our method. Finally, Section 4 summarizes the contribution of the paper.

2. The Proposed Method

Our proposed method consists of two stages. In the first stage, we perform *binarization*, where we convert continuous data into binary data. In the second stage, we perform feature selection to compress binarized data and remove unnecessary binary features in classification. We apply the Super-CWC algorithm to binary data in the second stage and obtain a minimal binary feature set that is consistent with given class labels.

[1]The Institute of Scientific and Industrial Research, Osaka University [2]National Institute of Information [3]JST PRESTO. Correspondence to: Yuka Yoneda <yoneda@ar.sanken.osaka-u.ac.jp>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

Algorithm 1 Binalization

$\boldsymbol{z}_i \leftarrow$ empty string for all $i \in \{1, 2, \ldots, N\}$
for all $j \in \{1, 2, \ldots, n\}$ **do**
  for all $i \in \{1, 2, \ldots, N\}$ **do**
    for all $k \in \{1, 2, \ldots, N\}$ **do**
      compute $z_i^{(j,k)}$ by Eq. (1)
    end for
    $\boldsymbol{z}_i \leftarrow$ concatenation of $\boldsymbol{z}_i$ and $(z_i^{(j,1)}, \ldots, z_i^{(j,N)})$
  end for
end for
$S \leftarrow \{(\boldsymbol{z}_1, y_1), (\boldsymbol{z}_2, y_2), \ldots, (\boldsymbol{z}_N, y_N)\}$
Output S

Input to the proposed method is a labeled continuous data set. Each data point consists of a pair $(\boldsymbol{x}_i, y_i)$ $(i = 1, 2, \ldots, N)$ of an n-dimensional vector $\boldsymbol{x}_i = (x_i^1, x_i^2, \ldots, x_i^n)$ and a class label $y_i \in \{0, 1\}$. For each feature $j \in \{1, 2, \ldots, n\}$, we describe $\boldsymbol{x}^j = (x_1^j, x_2^j, \ldots, x_N^j)$.

2.1. Binarization

In the binarization stage, we convert each continuous feature vector into a binary vector. A converted feature vector corresponds to an intersection of rectangular regions in the original feature space. Each binary feature corresponds to an interval and each binary value of the vector indicates whether or not the original continuous value belongs to the interval. We use each original continuous value as a threshold of binarization. In other words, x_i^j for each $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, n\}$ is converted into a binary vector of N components $(z_i^{(j,1)}, z_i^{(j,2)}, \ldots, z_i^{(j,N)})$ by binarization, where $z_i^{(j,k)}$ is defined as

$$z_i^{(j,k)} = \begin{cases} 1 & \text{if } x_i^j \text{ is smaller than } x_k^j, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We present our binarization algorithm in Algorithm 1, which works as follows: Input is a data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$ consisting of pairs of continuous feature vectors and class labels. We binarize each vector by Equation (1), that is, each $\boldsymbol{x}_i$ is converted into $\boldsymbol{z}_i$ such that

$$\boldsymbol{z}_i = \Big(z_i^{(1,1)}, \ldots, z_i^{(1,N)}, z_i^{(2,1)}, \ldots, z_i^{(2,N)}, \ldots, z_i^{(n,N)}, \ldots, z_i^{(n,N)} \Big). \quad (2)$$

Output of th binarization stage is a set of pairs $\boldsymbol{z}_i$ and y_i for all $i \in \{1, 2, \ldots, N\}$, that is, $S = \{(\boldsymbol{z}_1, y_1), (\boldsymbol{z}_2, y_2), \ldots, (\boldsymbol{z}_N, y_N)\}$.

2.2. Feature Selection by Super-CWC

In the second stage, we apply the feature selection algorithm Super-CWC (Shin et al., 2015) to binary data obtained by the discretization stage and delete unnecessary binary feature vectors for distinguishing class labels.

The Super-CWC algorithm performs supervised feature selection for binary data. It efficiently seeks feature combinations which are consistent with given class labels by binary search and can obtain necessary minimal feature combinations. Mathematically, for a subset of feature indices $I \subseteq \{1, 2, \ldots, nN\}$, I is *consistent* with class labels if the following condition is satisfied for all $i, i' \in \{1, \ldots, N\}$: $\boldsymbol{z}_i^j = \boldsymbol{z}_{i'}^j$ for all $j \in I$ implies $y_i = y_{i'}$. Super-CWC uses a function Bn such that $\mathrm{Bn}(I) = 0$ if I is consistent and $\mathrm{Bn}(I) = 1$ otherwise.

Super-CWC receives pairs $(\boldsymbol{z}_i, y_i)$ $(i = 1, \ldots, N)$ of an nN-dimensional binary vector $\boldsymbol{z}_i = (z_i^1, z_i^2, \ldots, z_i^{nN}) \in \{0, 1\}^{nN}$ and a class label $y_i \in \{0, 1\}$. Suppose that $[nN] = \{1, 2, \ldots, nN\}$ is the set of binarized features. Super-CWC searches a feature subset satisfying the following condition.

$$\min_{I \subseteq [nN], \mathrm{Bn}(I) = 0} |I|. \quad (3)$$

Note that the output of Super-CWC may be a local minimum solution. Therefore, an output is a minimal binary feature set which is consistent with given class labels. See the literature (Shin et al., 2015) for more detail about the algorithm of Super-CWC.

3. Experiments

In this section, we evaluate the proposed method using synthetic and real-world datasets. In Section 3.1, we describe experimental setting including the environment, a comparison partner, datasets, and evaluation measure. We show experimental results and discuss them in Section 3.2.

3.1. Experimental Setting

3.1.1. Experiment Environment

We performed all experiments on Windows10 Pro 64bit OS with a single processor of Intel Core i7-4790 CPU 3.60 GHz and 16GB of main memory. All experiments were conducted in Python 3. In our method, we used Scala implementation of Super-CWC algorithm provided by the authors[1].

3.1.2. Comparison Partner

As a comparison partner, we employed CART (Breiman et al., 1984), which is a popular decision tree based method

[1] https://github.com/tkub/scwc/

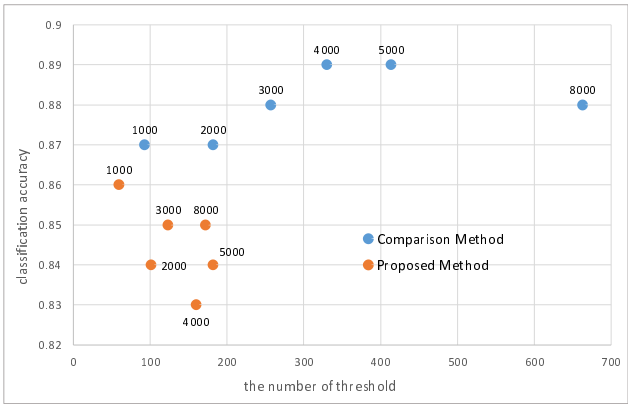

Figure 2. Results on synthetic dataset D_N, where $n = 2$ and N is from $1,000$ to $8,000$ (shown in numbers associated with points).

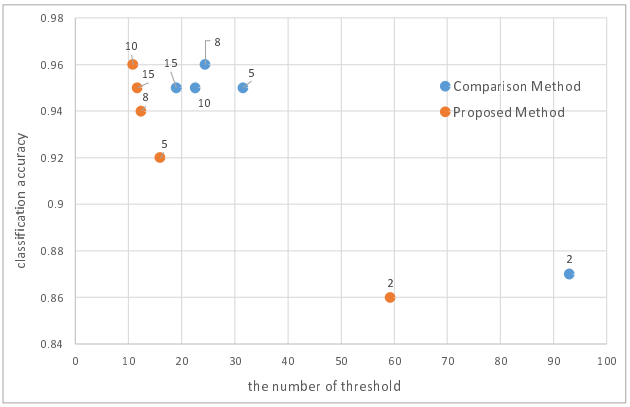

Figure 3. Results on synthetic dataset D_N, where $N = 1,000$ and n is from 2 to 15 (shown in numbers associated with points).

and the primary choice in scikit-learn (Pedregosa et al., 2011), one of the most popular machine learning libraries in Python. CART obtains explicit classification rules from continuous data.

3.1.3. Datasets

We used two types of synthetic datasets D_N and D_U. In the dataset D_N, data points with the class label 0 are generated from a n-dimensional normal distribution $(\mu_0, \sigma_0^2) = (0, 1)$, and those with the class label 1 are generated from another n-dimensional normal distribution $(\mu_1, \sigma_1^2) = (2, 1)$. In the dataset D_U, the first five features are generated from the above two n-dimensional normal distributions, while the other features are irrelevant (noises) to class labels, that is, they are generated from the m-dimensional uniform distribution between zero and one, resulting in $5 + m$-dimensional feature vectors.

For real-world data, we collected six datasets from UCI machine learning repository (Lichman, 2013) from small to

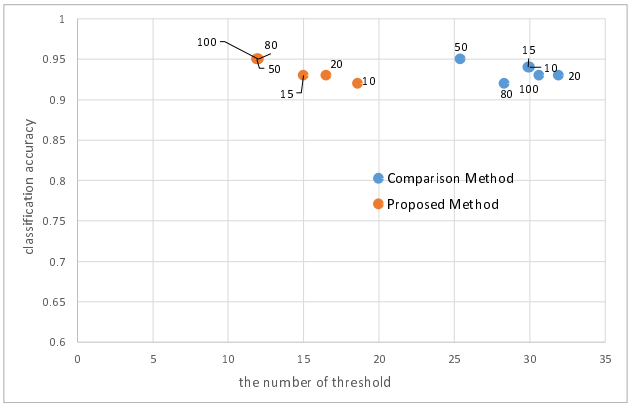

Figure 4. Results on synthetic dataset D_U, where $N = 1,000$ and m is from 10 to 100 (shown in numbers associated with points).

Table 1. Real-world datasets.

Name	# data points	# features
Parkinsons	197	23
Breast cancer	569	10
Vertebral	310	6
Wine quality	1600	12
CTG	2126	20
Seismic bumps	2584	25

large scale datasets (N is between about 200 to 2500), and used only continuous features. Their properties are summarized in Table 1.

3.1.4. Evaluation

We used 10-fold cross validation and obtained the average of classification accuracy. Moreover, to quantify the interpretability of classification rules, we used the number of thresholds of rules. Less thresholds mean simpler classification rules, leading to higher interpretability. The number of binary features in our method and that of nodes except for leaves in CART correspond to the number of thresholds. Thus a classifier is better if it has less thresholds while has high accuracy.

3.2. Results and Discussion

We plot results for synthetic datasets in Figures 2, 3, and 4, where the x-axis shows the number of thresholds and y-axis shows classification accuracy. First, in Figure 2, we used D_N and changed the number N of the data points from $1,000$ to $8,000$ while fixing the number of features $n = 2$. Although accuracy of our method is marginally lower than CART, where the difference is between 0.01 and 0.06, the number of thresholds is smaller than CART. When $N = 8,000$, the difference of the number of thresholds is

Table 2. Results on real-world datasets. Best scores are denoted in Bold.

Name	Number of thresholds		Accuracy	
	Our method	Decision tree	Our method	Decision tree
Parkinsons	21.4	**12.3**	0.82	**0.88**
Breast cancer	30.3	**27.9**	0.89	**0.93**
Vertebral	**26.4**	33.2	**0.78**	**0.78**
Wine quality	**40.2**	121.7	**0.87**	0.84
CTG	**38.4**	90.6	0.84	**0.86**
Seismic bumps	**89.5**	165.5	**0.92**	0.86

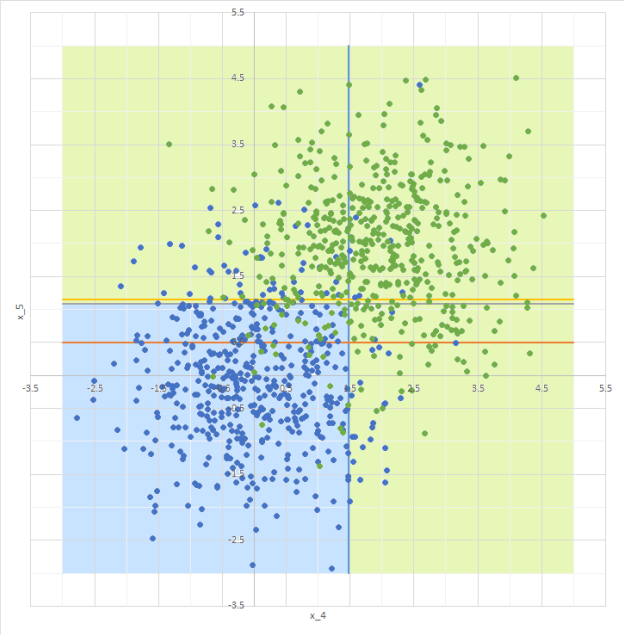

Figure 5. Classification rules obtained by our method.

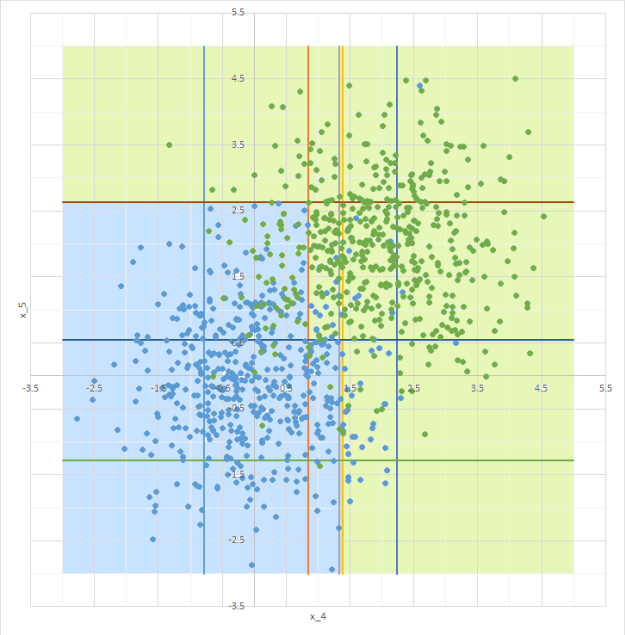

Figure 6. Classification rules obtained by CART.

about 500, which means that our method successfully obtains a simpler classification rule. Second, in Figure 3, we again used D_N and changed the number n of features from 2 to 15 while fixing the number of data points $N = 1,000$. Results show that our method can obtain simpler classification rules while keeping the same level of classification accuracy. In particular, when $n = 10$ or 15, accuracy of our method is better than CART. Third, in Figure 4, we used D_U and changed the number m of irrelevant features from 2 to 100 while fixing the number of data points $N = 1,000$. We can observe the same trend as before: our method consistently obtains simpler classification rules with less thresholds, while can achieve the same level of classification accuracy.

Results on real-world datasets are shown in Table 2, which again confirm that our method can obtain simpler classification rules that can achieve the same or better classification accuracy compared to the decision tree method. For the datasets Parkinsons and Breast cancer, the number of thresholds of our method is larger. This is why the number N of data points is too small, which makes it difficult to obtain good thresholds for binarization. This trend is consistent with results on synthetic data, where our method becomes superior when N is larger in terms of the number of thresholds in Figure 2.

To summarize, our results show that the proposed method can obtain simpler classification rules than a decision tree based method, CART, while attaining the same level of classification accuracy. The reason might be that we remove unnecessary binary features, that is, thresholds used in classification rules, by the feature selection algorithm Super-CWC.

3.3. Illustrative Example

We illustrate classification rules obtained by our method and CART on the synthetic dataset D_U with $m = 10$ and

$N = 1,000$ in Figures 5 and 6, respectively. In this example, the numbers of thresholds of our method and CART are 17 and 28, and the classification accuracies are 0.92 and 0.94, which means that our method obtains simpler rules with the almost same accuracy. In Figures 5 and 6, we plot two features out of five features generated from two normal distributions, where green dots show data points in the class 1 while blue dots are those in the class 0. In both figures, each line denotes a threshold of a classification rule and green and blue areas are regions classified as the classes 1 and 0, respectively. Theses figures demonstrate that our method learns simpler rules compared to CART, thanks to the feature selection stage by the Super-CWC algorithm.

4. Conclusions

In this paper, we have proposed a novel method that can find interpretable classification rules as intersections of rectangular regions. Our method first binarizes a given dataset and then performs supervised feature selection on the binarized data to remove unnecessary binary features. We have experimentally showed that we can obtain classification rules with less thresholds compared to CART, a popular decision tree based method, while attaining the same level of classification accuracy.

Our future work is to theoretically analyze the classification ability of our method and compare it to other classification methods, including decision tree methods.

References

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. *Classification and regression trees*. CRC press, 1984.

Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3 edition, 2011.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, pp. 436–444, 2015.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, pp. 2825–2830, 2011.

Quinlan, J. R. *Introduction of decision trees*. Machine Learning, 1986.

Quinlan, J. R. *Programs for Machine Learning*. Morgan Kaufmann, 1993.

Shin, K., Kuboyama, T., Hashimoto, T., and Shepard, D. Super-CWC and super-LCC: Super fast feature selection algorithms. In *Proceedings of 2015 IEEE International Conference on Big Data*, pp. 1–7, 2015.