TRANS-AM: Discovery Method of Optimal Input Vectors Corresponding to Objective Variables

Hiroaki Tanaka[1](✉), Yu Suzuki[1,2](✉), Koichiro Yoshino[1,3](✉), and Satoshi Nakamura[1,2](✉)

[1] Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Japan
{tanaka.hiroaki.sy2,ysuzuki,koichiro,s-nakamura}@is.naist.jp

[2] Data Science Centre, Nara Institute of Science and Technology, Ikoma, Japan

[3] PRESTO: Fundamental Information Technologies toward Innovative Social System Design. Japan Science and Technology Agency, Kyoto, Japan

Abstract. In various fields, ensemble models by supervised learning are effective, but the models cannot tell us how to modify the input vector so that we will increase the objective variable more than a given threshold or decrease it less than the threshold. In this paper, we propose a method, TRANS-AM, that can discover an input vector satisfying the condition of changing of the objective variable in regression problems by using a property of regression tree. The regression tree splits input space into subspaces. There are subspaces with corresponding objective variables satisfying such a condition. By transforming the input vector to new input vectors belonging to one of the subspaces, we can discover a new input vector whose distance from the original input vector is minimum by satisfying the condition to change the objective variable. The reason for "minimum" is the cost—if the new input vector is far from the original one, we need the significant cost to modify the original input vector to the new one. We evaluated the proposed method through numerical simulations and investigated that the proposed method works well; the ratio of the number of discovered input vectors satisfying the condition per the number of discovered input vectors is 60% for the datasets generated through logistic function.

1 Introduction

As most of the researches in data mining and machine learning has focused on the accuracy, efficiency, and robustness of different techniques, little effort has been made for "actionable knowledge extraction" from advanced machine learning models. Here, "actionable knowledge extraction" means finding how to change input vectors to improve the objective variables in supervised learning: improving the objective variable in regression tasks. However, in the real world, we often

H. Tanaka—Presently with Research Laboratory, NTT DOCOMO Inc.

C. Ordonez and L. Bellatreche (Eds.): DaWaK 2018, LNCS 11031, pp. 216–228, 2018.
https://doi.org/10.1007/978-3-319-98539-8_17

face the difficulty of extracting such knowledge. In other words, we cannot obtain the knowledge to answer the research question "How do we modify the input vectors to increase the objective variables greater than a given threshold or decrease them less than the threshold with minimum effort?"

Let us consider a running example. We are strategists of a soccer team. It is supposed that the winning percentage is predicted by random forest regression whose objective variables are abilities of each position: speed, acceleration, dash speed, and so on. If we can acquire the desirable input variables corresponding to winning percentage which is upper than a given threshold—for example, we want to keep the winning percentage upper than 80%, the threshold is 80; we can plan the training so that the abilities of players will satisfy the desirable input variables. In addition, if the desirable input variables corresponding to the winning percentage which is upper than the threshold is far from the original input variables, we cannot develop the good plan—it is difficult to improve the players' abilities exponentially. Therefore, the requirement is rewritten as "We want to know the desirable abilities of each player and the abilities are nearest to original abilities of players."

In this paper, we propose a method named, the Transforming-feAture Method (TRANS-AM) to answer the research question in a regression task. This method enables us to modify the input vector adequately to increase or decrease the objective variables by adding a small positive number ε to each input variable in the original input variables. With the property of a regression tree—it is supposed that the objective variable is predicted by random forest regression [1]—, splitting the input space into subspaces, allows us to determine the input vector whose objective variable improved. By transforming the input vector to new input vectors belonging to one of the subspaces, we discover a new input vector whose distance from the original input vector is minimum in those new vectors. The basic idea of the TRANS-AM is based on the method proposed by Tolomei *et al.* [2]. Their method is built for classification tasks, and we expanded the method for regression tasks. We also relax a restriction of a regression tree which is supposed in the previous study by Tolomei *et al.* [2]. Considering the previous running example, the TRANS-AM enables us to plan the ideal training.

The contributions of this paper are as follows.

- We expanded the classification task discussed in Tolomei *et al.* [2] to the regression task.
- We relaxed a restriction which is assumed in Tolomei *et al.* [2]: once we use the input variable x_i for a branch of tree we cannot use it for other branches.

The second contributions allows us to use more complex regression trees for the random forest more than that used in Tolomei *et al.* [2].

2 Related Work

We first discuss earlier studies on extracting actionable knowledge. Cao *et al.* [3] and Robert *et al.* [4] focused on the development of effective interestingness metrics. Liu *et al.* [5,6] proposed methods for pruning and summarizing the learnt rules, as well as matching rules by similarity. Cao et al. [7,8] proposed a data mining method which is a paradigm shift from a research-centred discipline to a practical tool for actionable knowledge. All the above methods depend on the domains of datasets, but our proposed method does not.

We now discuss tree-based methods. Du *et al.* [9], Karim and Rahman [10], and Yang *et al.* [11,12] discussed post-proceeding methods specifically tailored to decision trees. It is true that a decision tree is interpretable; therefore, we can find a modification approach to improve the objective variable. However, a decision tree's prediction precision is not good [13]. Our proposed method, on the other hand, can use random forest whose prediction precision is better than that of a decision tree.

Finally, Cui *et al.* [14] proved that the optimal action extraction (OAE) problem similar to the problem we solve in Sect. 3 is generally NP-hard by reducing it to DNF-MAXSAT [15] and formulated the problem in an integer linear programming formulation, which has been efficiently solved using current packages with state-of-the-art solvers such as CPLEX [16]. Tolomei *et al.* [2] developed a method of solving the OAE problem with an ε-satisfactory instance, which is explained in Sect. 3. We call this method as actionable feature tweaking (AFT). As mentioned in Sect. 1, the AFT requires the restriction: once we use the input variable x_i for a branch, we cannot use it for other branches. By this restriction, the modification of input vectors is simplified. As we relax the restriction, we change the modification approach, i.e. the approach to developing an ε-satisfactory instance.

3 TRANS-AM: Proposed Method

In this section, we explain the TRANS-AM for feature transformation to increase objective variables more than a given threshold or decrease them than the threshold by expanding AFT. Actionable feature tweaking is used to change the label of an objective variable, i.e. the task addressed in AFT is *classification*. We expanded AFT to change the objective variable to *regression* and liberalize one assumption of AFT regarding the root-to-leaf paths of each regression tree. We published the source of TRANS-AM on https://github.com/setten-QB/TRANS-AM.git

3.1 Notation

Let $\mathcal{X} \subset \mathbb{R}^d$ be an input space and suppose that each $\boldsymbol{x} \in \mathcal{X}$ is associated with an objective variable $y \in \mathcal{Y} \subset \mathbb{R}$. We assume there is an unknown target function $f : \mathcal{X} \to \mathcal{Y}$. Most machine learning methods learn the function $\hat{f} \approx f$ from dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$. Specifically, $\hat{f}$ is the estimate that best approximates f on $\mathcal{D}$, according to a specific loss function ℓ. The ℓ measures the error between predicted and observed values.

The interpretability of $\hat{f}$ depends on the hypothesis space in which $\hat{f}$ was selected. In this study, we focused on random forest regression. Random forest regressor $\mathcal{T}$ consists of K regression trees $T_1, \cdots, T_K$. We represent the estimate of each T_k, $k = 1, \cdots, K$ as $\hat{h}_k$. Then the estimate of $\mathcal{T}$ is calculated by the sample mean of $\hat{h}_k$.

Regression tree T_k splits the input space $\mathcal{X}$ into subspaces as

$$\mathcal{X} = \mathcal{X}_{k,1} \oplus \mathcal{X}_{k,2} \oplus \cdots \oplus \mathcal{X}_{k,M}, \tag{1}$$

and $\gamma_{k,m}$ corresponds to the area $\mathcal{X}_{k,m}$. Then the prediction with T_k is calculated by

$$\hat{h}_k(\boldsymbol{x}) = \sum_{m=1}^{M} \gamma_{k,m} 1[\boldsymbol{x} \in \mathcal{X}_{k,m}], \quad 1[\boldsymbol{x} \in \mathcal{X}_{k,m}] = \begin{cases} 1 & \boldsymbol{x} \in \mathcal{X}_{k,m} \\ 0 & \boldsymbol{x} \notin \mathcal{X}_{k,m} \end{cases}. \tag{2}$$

where $1[\boldsymbol{x} \in \mathcal{X}_{k,m}]$ means the indicator function which is defined as

$$1[\boldsymbol{x} \in \mathcal{X}_{k,m}] = \begin{cases} 1 & \boldsymbol{x} \in \mathcal{X}_{k,m} \\ 0 & \boldsymbol{x} \notin \mathcal{X}_{k,m} \end{cases}. \tag{3}$$

3.2 Split Input Space with Regression Tree

Suppose that we are given the trained random forest regressor $\mathcal{T}$ and $\boldsymbol{x}$ satisfying $f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) < t_0$, where t_0 is a hyperparameter meaning a lower threshold. Our aim is to transform $\boldsymbol{x}$ to $\boldsymbol{x}'$, which satisfies $f(\boldsymbol{x}') \geq t_1$, where t_1 is also a hyperparameter meaning an upper threshold.

We assume that Assumption 1 holds for any fixed regression tree T_k.

Assumption 1. *For any fixed regression tree T_k of the random forest $\mathcal{T}$, 4 holds.*

$$\exists \mathcal{X}_{k,m} \subset \mathcal{X} \quad s.t. \quad \forall \boldsymbol{x} \in \mathcal{X}_{k,m},\ \hat{h}_k(\boldsymbol{x}) = \gamma_{k,m} \geq t_1 \tag{4}$$

By this assumption, we can select the subspace $\mathcal{X}_{k,m}^{t_1}$ whose $\gamma_{k,m}$ satisfies $\gamma_{k,m} \geq t_1$, and $\mathcal{X}_{k,m}^{t_1}$ is written as

$$\mathcal{X}_{k,m}^{t_1} = \prod_{i=1}^{d} \left[\theta_i^{\text{low}}, \theta_i^{\text{upp}}\right], \quad \theta_i^{\text{low}}, \theta_i^{\text{upp}} \in \mathbb{R} \cup \{-\infty, \infty\}. \tag{5}$$

In (5), θ_i^{low} and θ_i^{upp} are decided by the random forest algorithm. In AFT, we have to assume Assumption 2.

Assumption 2 (In AFT). *The path $p_{k,m}$ of regression tree T_k from root to m-th leaf is represented as*

$$p_{k,m} = \left\{ \left(x_1 \gtreqless \theta_1\right), \left(x_2 \gtreqless \theta_2\right), \cdots, \left(x_d \gtreqless \theta_d\right) \right\}. \tag{6}$$

Assumption (6) means that for all i either $\theta_i^{\text{low}} = -\infty$ or $\theta_i^{\text{upp}} = \infty$ holds. However, with the TRANS-AM, we do not adopt Assumption 2 because in many actual cases the learned regression tree does not satisfy this assumption. Of course we can build a regression tree by satisfying Assumption 2, but it is a little messy and sacrifices prediction flexibility.

3.3 ε-Satisfactory Instance

Let $\mathscr{X}_k$ be the family of all $\mathcal{X}_{k,m}^{t_1}$ in T_k;

$$\mathscr{X}_k = \{ \mathcal{X}_{k,m} \mid \forall \boldsymbol{x} \in \mathcal{X}_{k,m},\, \hat{h}_k(\boldsymbol{x}) = \gamma_{k,m} \geq t_1 \}, \tag{7}$$

where $\mathcal{X}_{k,m}$ satisfying $\forall \boldsymbol{x} \in \mathcal{X}_{k,m}$, $\hat{h}_k(\boldsymbol{x}) = \gamma_{k,m} \geq t_1$ is $\mathcal{X}_{k,m}^{t_1}$. Then, $|\mathscr{X}_k|$ is often greater than 1. For all subspaces $\mathcal{X}_{k,m}^{t_1} \in \mathscr{X}$, we build the ε-satisfactory instances $\boldsymbol{x}_{k(\varepsilon)}^{t_1}$ as following.

$$\boldsymbol{x}_{k(\varepsilon)}^{t_1}[i] = \begin{cases} \theta_i^{\text{upp}} - \varepsilon & \theta_i^{\text{low}} = -\infty \\ \theta_i^{\text{low}} + \varepsilon & \theta_i^{\text{upp}} = \infty \\ \frac{\theta_i^{\text{low}} + \theta_i^{\text{upp}}}{2} & \text{otherwise} \end{cases} \tag{8}$$

In (8), ε means the distance between $\boldsymbol{x}_{k(\varepsilon)}^{t_1}$ and the boundaries of subspace, i.e., the position of $\boldsymbol{x}_{k(\varepsilon)}^{t_1}$ is determined by ε if $\theta_i^{\text{low}} = -\infty$ or $\theta_i^{\text{upp}} = \infty$.

The ε-satisfactory instance defined by (8) belongs to $\mathcal{X}_{k,m}^{t_1} \in \mathscr{X}$. If $\boldsymbol{x}_{k(\varepsilon)}^{t_1}[i] = \theta_i^{\text{upp}} - \varepsilon$, the i-th element of $\boldsymbol{x}_{k(\varepsilon)}^{t_1}$ is the point transformed by $-\varepsilon$ from θ_i for the negative direction parallel to the x_i axis (or vice versa). If $\boldsymbol{x}_{k(\varepsilon)}^{t_1}[i] = \left(\theta_i^{\text{low}} + \theta_i^{\text{upp}}\right)/2$, the i-th element of $\boldsymbol{x}_{k(\varepsilon)}^{t_1}$ is the centre of interval $\left[\theta_i^{\text{low}}, \theta_i^{\text{upp}}\right]$. In any case $\boldsymbol{x}_{k(\varepsilon)}^{t_1} \in \mathcal{X}_{k,m}^{t_1}$ holds; therefore, $\hat{h}\left(\boldsymbol{x}_{k(\varepsilon)}^{t_1}\right) \geq t_1$ also holds.

3.4 Feature Transformation

Let $\mathcal{X}_{\cdot,\cdot}^{t_1}$ be the set of all ε-satisfactory instances. More specifically, $\mathcal{X}_{\cdot,\cdot}^{t_1}$ is written by

$$\mathcal{X}_{\cdot,\cdot}^{t_1} = \bigcup_{k=1}^{K} \bigcup_{m:\mathcal{X}_{k,m}^{t_1} \in \mathscr{X}} \boldsymbol{x}_{k(\varepsilon)}^{t_1}, \tag{9}$$

where $\boldsymbol{x}_{k(\varepsilon)}^{t_1}$ depends on subspace $\mathcal{X}_{k,m}^{t_1}$. Therefore, $\bigcup_{m:\mathcal{X}_{k,m}^{t_1} \in \mathscr{X}} \boldsymbol{x}_{k(\varepsilon)}^{t_1}$ means the set of all the ε-satisfactory instances made using T_k. Then the transformed input vector we want is given by solving the following optimization problem.

$$\boldsymbol{x}' = \mathop{\arg\min}_{\boldsymbol{x}_{k(\varepsilon)}^{t_1} \in \mathcal{X}_{\cdot,\cdot}^{t_1} \mid \hat{f}\left(\boldsymbol{x}_{k(\varepsilon)}^{t_1}\right) \geq t_1} \delta\left(\boldsymbol{x}, \boldsymbol{x}_{k(\varepsilon)}^{t_1}\right) \tag{10}$$

In (10), cost function δ means the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_{k(\varepsilon)}^{t_1}$. With this δ, we choose optimal vector $\boldsymbol{x}'$, i.e. the selected $\boldsymbol{x}'$ is nearest to the original input vector. As we explained in Sect. 1, Cui *et al.* [14] proved that the OAE problem, which is essentially identical to (10), is generally NP-hard, and the TRANS-AM translates the OAE problem into a closed-form integer linear programming (ILP) problem, which can be efficiently solved by off-the-shelf ILP solvers. On the other hand, Tolomei *et al.* [2] introduced an algorithm to obtain $\boldsymbol{x}'$ by using an ε-satisfactory instance.

In this paper, we used an expanded algorithm of Tolomei *et al.* [2] to discovery $\boldsymbol{x}'$. We show the algorithm for finding $\boldsymbol{x}'$ in Algorithm 1. This algorithm seeks $\boldsymbol{x}'$ which is solution of the following optimization problem:

$$\boldsymbol{x}' = \underset{\boldsymbol{x}_{k(\varepsilon)}^{t_1} \in \mathcal{X}_{\cdot,\cdot}^{t_1}}{\arg\min}\ \delta\left(\boldsymbol{x}, \boldsymbol{x}_{k(\varepsilon)^{t_1}}\right) \tag{11}$$

In (11), we dropped the condition $\hat{f}\left(\boldsymbol{x}_{k(\varepsilon)}^{t_1}\right) \geq t_1$ of (10), because the condition is implicitly satisfied by definition of ε-satisfactory instance. Algorithm 1 often returns $\boldsymbol{x}' = \boldsymbol{x}$ because sometimes all the ε-satisfactory instances $\boldsymbol{x}_{j(\varepsilon)}^{t_1}$ built with Algorithm 1 do not satisfy the 8-th line if-statement $\hat{f}\left(\boldsymbol{x}_{j(\varepsilon)}^{t_1}\right) \geq t_1$. Hence we should evaluate the TRANS-AM carefully in Sect. 4.

Algorithm 1. Algorithm of TRANS-AM

```
Require: $\mathcal{T} = \{T_1, T_2, \cdots, T_K\}$, thresholds $t_0, t_1$, input vector $\boldsymbol{x}$ such that $f(\boldsymbol{x}) < t_0$,
    cost function $\delta$, and $\varepsilon > 0$
Ensure: $\boldsymbol{x}'$ satisfying $\hat{f}(\boldsymbol{x}') \geq t_1$
 1: $\boldsymbol{x}' \leftarrow \boldsymbol{x}$
 2: $\delta_{\min} \leftarrow \infty$
 3: for $k = 1, 2, \cdots, K$ do
 4:     if $\hat{f}(\boldsymbol{x}) < t_1 \land \hat{h}_k(\boldsymbol{x}) < t_1$ then
 5:         make $\mathscr{X}_k$
 6:         for $\mathcal{X}_{k,m}^{t_1} \in \mathscr{X}_k$ do
 7:             build $\varepsilon$-satisfactory instance $\boldsymbol{x}_{j(\varepsilon)}^{t_1}$
 8:             if $\hat{f}\left(\boldsymbol{x}_{j(\varepsilon)}^{t_1}\right) \geq t_1$ then
 9:                 if $\delta\left(\boldsymbol{x}, \boldsymbol{x}_{j(\varepsilon)}^{t_1}\right) < \delta_{\min}$ then
10:                     $\boldsymbol{x}' \leftarrow \boldsymbol{x}_{j(\varepsilon)}^{t_1}$
11:                     $\delta_{\min} \leftarrow \delta\left(\boldsymbol{x}, \boldsymbol{x}_{j(\varepsilon)}^{t_1}\right)$
12:                 end if
13:             end if
14:         end for
15:     end if
16: end for
17: return $\boldsymbol{x}'$
```

4 Numerical Simulation and Evaluation

In this section, we explain the results of numerical simulations. The aim with TRANS-AM is to transform the input vector $\boldsymbol{x}$ with $\hat{f}(\boldsymbol{x}) < t_0$ to $\boldsymbol{x}'$ satisfying

$f(\boldsymbol{x}') \geq t_1$. If the random forest estimates the unknown target function f, Algorithm 1 can return $\boldsymbol{x}'$ such that $f(\boldsymbol{x}') \geq t_1$. However, in practice the random forest cannot estimate f completely, so the vector $\boldsymbol{x}'$ yielded from Algorithm 1 does not always satisfy $f(\boldsymbol{x}') \geq t_1$, i.e. Algorithm 1 sometimes yields $\boldsymbol{x}'$ satisfying not $f(\boldsymbol{x}') \geq t_1$ but $\hat{f}(\boldsymbol{x}) \geq t_1$. Therefore, we should evaluate how many vectors yielded from Algorithm 1 satisfy $f(\boldsymbol{x}') \geq t_1$. For such an evaluation, we should know the target function f; hence, we evaluated the TRANS-AM not through application for real datasets but numerical simulations.

As we mentioned in Sect. 3, Algorithm 1 often returns $\boldsymbol{x}' = \boldsymbol{x}$ Therefore, we use (18) as an indicator for evaluating this problem.

4.1 Experimental Setting

We evaluated the TRANS-AM with artificial data. The artificial datasets were generated in the following steps, where $\boldsymbol{1}_d$ is the d-dimensional vector whose components are 1, I_d is the $d \times d$ diagonal matrix whose diagonal components are 1, and $\mathcal{N}_d(\boldsymbol{1}_d, I_d)$ means the d-dimensional Gaussian distribution whose mean vector is $\boldsymbol{1}_d$ and covariance matrix is I_d.

Step 1. generate $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N \overset{\text{i.i.d.}}{\sim} \mathcal{N}_d(\boldsymbol{1}_d, I_d)$.
Step 2. make independent variable $y_1, y_2, \cdots, y_N$ by $y_n = f(\boldsymbol{x}_n) + \eta_n$, $\eta_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$

Note that f in the above steps is the same as the unknown target function in Sect. 3. In our simulation, we used the following functions as $f(\boldsymbol{x})$.

$$f_1(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{x} \tag{12}$$

$$f_2(\boldsymbol{x}) = \sin\left(\boldsymbol{a}^T \boldsymbol{x}\right) \tag{13}$$

$$f_3(\boldsymbol{x}) = \sum_{i=1}^{n} a_i \sin x_i \tag{14}$$

$$f_4(\boldsymbol{x}) = \exp\left(\boldsymbol{a}^T \boldsymbol{x}\right) \tag{15}$$

$$f_5(\boldsymbol{x}) = \frac{1}{1 + \exp\left(\boldsymbol{a}^T \boldsymbol{x}\right)} \tag{16}$$

In functions (12)–(16), $\boldsymbol{a} \sim \mathcal{N}(\boldsymbol{0}, I)$ and a_i is the i-th element of $\boldsymbol{a}$. The TRANS-AM has the parameters t_0, t_1 and ε. Parameters t_0 and t_1 are determined by analysts according to their aim, and ε should be turned using some kind of data-driven method. The following steps comprise the simulation process.

step 1. split the dataset $\mathcal{D}$ into training set $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N_0}$ and test set $\mathcal{D}_{\text{test}} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N_1}$, where $N = N_0 + N_1$
step 2. let t_1 be the p_{upp} percentile point of $y \in \mathcal{D}_{\text{train}}$ and let t_0 be the p_{low} percentile point of $y \in \mathcal{D}_{\text{train}}$.
step 3. fix ε to one of candidates of ε and train the random forest regressor using $\mathcal{D}_{\text{train}}$.

step 4. choose the input vectors in $\mathcal{D}_{\text{test}}$ whose objective variables are less than t_0.
step 5. transform the input vectors to the new input vectors $\boldsymbol{x}'$ with the TRANS-AM.
step 6. evaluate the TRANS-AM by using three criterion that are shown in (17), (18), and (19).

Score P represents how many input vectors are transformed using the TRANS-AM regardless of whether $\boldsymbol{x}'$ satisfies $f(\boldsymbol{x}') \geq t_1$, score Q indicates how many input vectors are modified per number of input vectors, and score R represents how many modified input vectors $\boldsymbol{x}'(\neq \boldsymbol{x})$ satisfy $f(\boldsymbol{x}') \geq t_1$ per number of changed input vectors. Algorithm 1 often yields the same vector as an input vector. Therefore, we evaluated how many yielded vectors satisfy our purpose $f(\boldsymbol{x}') \geq t_1$ per number of transformed vectors with the score R.

$$P = \frac{|\{\, \boldsymbol{x}' \mid f(\boldsymbol{x}') \geq t_1 \,\}|}{\left|\{\, \boldsymbol{x} \mid \hat{f}(\boldsymbol{x}) < t_0 \,\}\right|} \tag{17}$$

$$Q = \frac{|\{\, \boldsymbol{x}' \mid \boldsymbol{x}' \neq \boldsymbol{x} \,\}|}{\left|\{\, \boldsymbol{x} \mid \hat{f}(\boldsymbol{x}) < t_0 \,\}\right|} \tag{18}$$

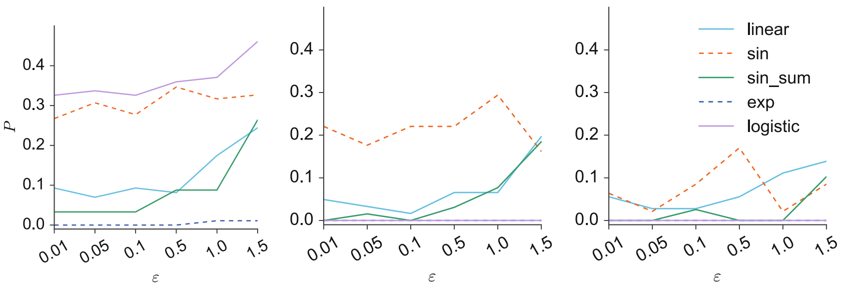
$$R = \frac{|\{\, \boldsymbol{x}' \mid f(\boldsymbol{x}') \geq t_1 \,\}|}{|\{\, \boldsymbol{x}' \mid \boldsymbol{x}' \neq \boldsymbol{x} \,\}|} \tag{19}$$

Among these three scores, a relationship $P = QR$ holds. Score P is most noticeable score and scores Q and R construct P. We simulated all combinations of $\varepsilon \in \{\, 0.01, 0.05, 0.1, 0.5, 1, 1.5 \,\}$, $N_0 = 1000$, $N_1 = 250$ and $d = 50$.

4.2 Result and Consideration

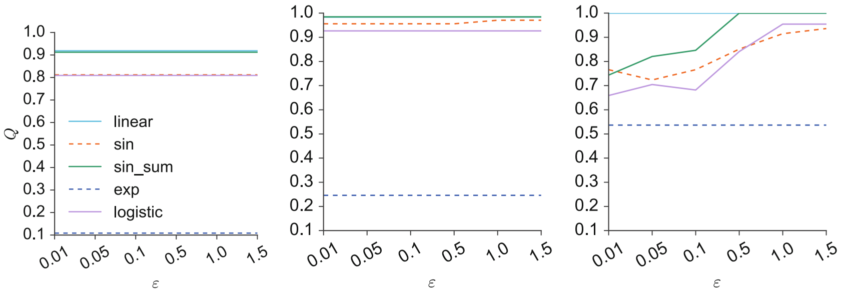
We show the simulation results in Figs. 1, 2 and 3. Figure 1 shows the relationship between ε and score P, Fig. 2 shows the relationship between ε and score Q, and Fig. 3 shows the relationship between ε and score Q. In the each figure, *linear, sin, sinsum, exp*, and *logistic* mean the datasets generated by (12), (13), (14), (15), and (16), respectively.

As shown in Fig. 1, for $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, the dataset generated by (16) shows the best score P. In addition, (13) shows the second-best score P. This is because (16) and (13) are upper bounded, i.e. $\exists K$: const. s.t. $\forall \boldsymbol{x} \in \mathbb{R}$, $f(\boldsymbol{x}) < K$. As we explained in Sect. 3, the regression tree splits the input space into subspaces $\{\, \mathcal{X}_m \,\}_{m=1}^{M}$ then corresponds γ_m with $\mathcal{X}_m$. The precisions of approximating the functions that are not upper bounded with random forest become worse due to this approximation. For example, suppose that we approximate the function $g(x) = \exp(x)$ and can use the regression tree. Then the input space is divided into M subspaces $\mathcal{X}_1, \cdots, \mathcal{X}_M$. The prediction value γ_m of $x \in \mathcal{X}_m$ is generally $(1/|\,\{\, x \in \mathcal{X}_m \,\}\,|) \sum_{x \in \mathcal{X}_m} x$, where x is the training sample. Therefore, the difference $\|f(x) - \gamma_m\|$ increases as x becomes larger.



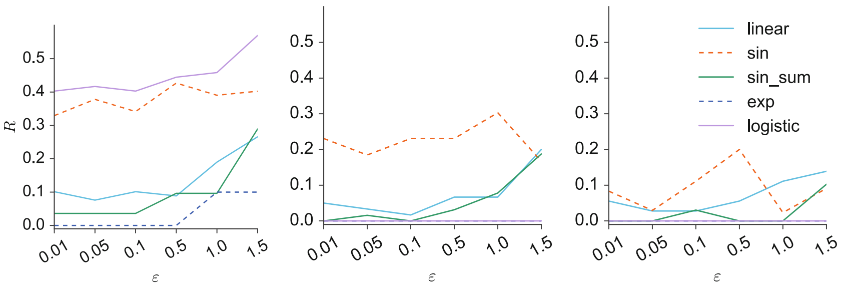
(a) $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$ (b) $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$ (c) $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

Fig. 1. Relationships between ε and P



(a) $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$ (b) $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$ (c) $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

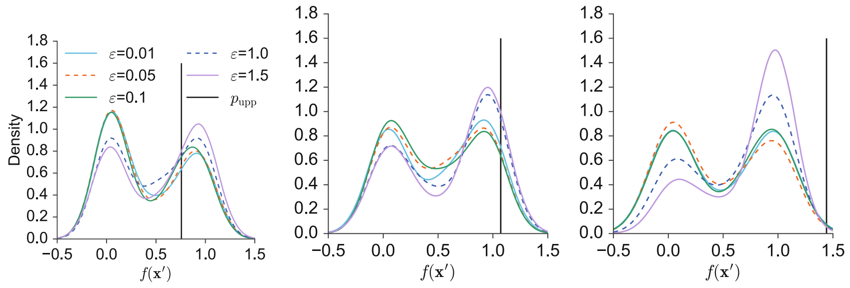
Fig. 2. Relationships between ε and Q



(a) $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$ (b) $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$ (c) $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

Fig. 3. Relationships between ε and R

On the other hand, in Fig. 1(b) and (c), the score P for the dataset generated by (16) becomes worse than that in the case of Fig. 1(a). The distribution of $f(\boldsymbol{x}')$ is illustrated in Fig. 4. As shown in Fig. 4, the threshold is pulled in the positive direction as percentile point becomes larger. Essentially, the objective variable does not appear in the region upper than 1 due to the range of (16);



(a) $(p_{\mathrm{low}}, p_{\mathrm{upp}}) = (40, 60)$ (b) $(p_{\mathrm{low}}, p_{\mathrm{upp}}) = (30, 70)$ (c) $(p_{\mathrm{low}}, p_{\mathrm{upp}}) = (20, 80)$

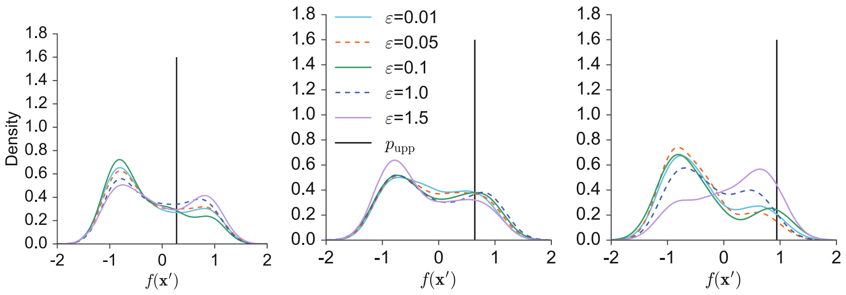
Fig. 4. Distributions of the objective variables generated by (16) and the threshold

we call the range of $f(x)$ "essential range of the objective variable"—the noise η affects the range of the objective variable. The reason why the threshold in the centre and right of Fig. 4 is greater than 1 is affection of noise η. In the case of Fig. 4(b) and (c), the setting of simulation—we aim to transform $\boldsymbol{x}$ into $\boldsymbol{x}'$ satisfying $f(\boldsymbol{x}') \geq p_{\mathrm{upp}}$—is essentially impossible, because the objective variable does not become greater than p_{upp}. Therefore, the results of Fig. 4(b) and (c) are not strange. Videlicet, the settings of the centre and right of Fig. 5 are not suitable, i.e. it is essentially impossible to obtain the transformed input vector $\boldsymbol{x}'$ satisfying $f(\boldsymbol{x}') \geq t_{\mathrm{upp}}$ because the threshold is out of the essential range of the objective variable.

As shown in Fig. 2(a) and (b), each score R is higher than 0.8, except for *exp*. These results mean that for the datasets generated by (15), Algorithm 1 cannot transform $\boldsymbol{x}$ into $\boldsymbol{x}'$. This is also because random forest cannot approximate Exponential Function (15) well. Of course, other functions (12) and (14) are not upper bounded. However, Exponential Function (15) diverges to infinity earlier than (12) and (14). For $f(x) = \exp(ax)$, the difference between $f(x)$ and $f(x + c)$, where c is a very small positive integer, is larger than that for $f(x) = ax$. Therefore, the approximation precision for the dataset generated by (15) becomes worse than that for the other. If the prediction precision is very bad, the condition $\hat{f}\left(\boldsymbol{x}^{t_1}_{j(\varepsilon)}\right) \geq t_1$ in Algorithm 1 cannot be satisfied, i.e. $\boldsymbol{x}$ is yielded using Algorithm 1. Therefore, we can conclude that for the dataset whose objective variable is generated by the exponential function of the input variable of the TRANS-AM cannot find the modified $\boldsymbol{x}'$. In the right figure of Fig. 2, turning the ε well, the score Q become higher than 0.8.

As shown in the Fig. 3(a), the score Rs for the datasets generated by (13) and (16) can be greater than 0.4 by turning ε for $p_{\mathrm{low}} = 40$, $p_{\mathrm{upp}} = 60$. In Fig. 3(b) and 1(c), however, the score R for the datasets generated by (16) one of the lowest. The reason is referred to in the discussion of Fig. 1. As p_{low} becomes smaller and p_{upp} becomes larger, the score R decrease.

Figure 5 illustrates the reason that the TRANS-AM performs well for the dataset generated by (13). The thresholds p_{upp} for *sin* are not pulled in the



(a) $(p_{\mathrm{low}}, p_{\mathrm{upp}}) = (40, 60)$ (b) $(p_{\mathrm{low}}, p_{\mathrm{upp}}) = (30, 70)$ (c) $(p_{\mathrm{low}}, p_{\mathrm{upp}}) = (20, 80)$

Fig. 5. Distributions of the objective variables generated by (13) and the threshold

positive direction so strongly, as for *logistic*, i.e. the threshold appears in the essential range of the objective variable. Therefore, the TRANS-AM performed better for the dataset generated by (13) than for the other datasets.

In all of that, the TRANS-AM does not work if the following Case 1 or 2 holds:

Case 1: the random forest cannot approximate the target function, e.g. the objective variables are generated from an upper unbound function;
Case 2: the threshold is out of the essential range of the objective variable.

The case 1 corresponds to the case of exp (see Figs. 1, 2 and 3) and the case 2 corresponds to the case of sin and logistic (see the centre and right of Fig. 1 and 3). However, if neither of the two cases holds, TRANS-AM will work well—we can obtain the input vectors satisfying that the objective variables are greater than the threshold from the candidates of input vectors by the score $P \geq 45$.

5 Conclusion

We proposed the TRANS-AM for discovering a new input vector to increase the objective variable greater than a given threshold or decrease it than the threshold with minimizing δ in a regression task. We relaxed the restrictions assumed by Tolomei *et al.* [2] because the restrictions were not reasonable for random forest. As we discussed in the introduction, we are often faced with the question "How do we modify the input vector to increase the objective variable greater than the given threshold or decrease it than the threshold with minimum effort?" The TRANS-AM is an answer to this question and we have disclosed the situation that the method works well.

In the TRANS-AM, we generated candidates of the new input vector by ε-satisfactory instances. The ε-satisfactory instances satisfy the condition, i.e. the objective variables corresponding to the ε-satisfactory instances are greater than the threshold. Then we select the new input vector which minimizes the distance between the original input vector and new one—we select the new input vector by (11).

Considering the use case of the TRANS-AM, we have to pay attention to what kind of input variables are contained in the input vector. The transformation by TRANS-AM affects all of the input variables generally, because of (8). Therefore, in the use case, we have to use only input variables which can be changed. In the case of example referred in Sect. 1—we are strategists of a soccer team, we can use the *energy*, *speed* of players. However, we cannot use the body size of the players, because we cannot control their values.

We evaluated the TRANS-AM by moving ε and the thresholds—p_{low} and p_{upp}—in Sect. 4. Then we found that for the dataset whose objective variables are generated by the sin formula, the TRANS-AM shows good precision; score P and score R are maximum or pre-maximum in each case of Figs. 1 and 3. On the other hand, for the dataset whose objective variables are generated by (16), the TRANS-AM does not work well except when $(p_{\mathrm{low}}, p_{\mathrm{upp}}) = (40, 60)$. In such a situation, it is hard for TRANS-AM to find the transformed input vector satisfying the condition. These considerations are summarized as the Case 1 and Case 2. If we avoid the two cases, i.e. we can approximate the target function by the random forest and the threshold is not out of the essential range of the objective variable; we obtain the transformed input vector efficiently by using TRANS-AM.

For future works, we plan to extend the TRANS-AM for gradient boosting [17] and XGBoost [18]—these additive tree models are effective predictors for various fields [13,19–21]. TRANS-AM works well for datasets whose objective variables are generated by the sin formula but does not work well for datasets whose objective variables are generated by an upper unbounded formula (e.g. exp formula). Therefore, a method that works well for upper unbounded datasets is required for discovering new input vectors whose objective variables are larger than a given threshold.

Acknowledgement. This research was partially supported by NAIST Big Data Project.

References

1. Breiman, L.: Random forests. Mach Learn. **45**(1), 5–32 (2001)
2. Tolomei, G., Silvestri, F., Haines, A., Lalmas, M.: Interpretable predictions of tree-based ensembles via actionable feature tweaking. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 465–474. ACM (2017)
3. Cao, L., Luo, D., Zhang, C.: Knowledge actionability: satisfying technical and business interestingness. IJBIDM **2**, 496–514 (2007)
4. Hilderman, R.J., Hamilton, H.J.: Applying objective interestingness measures in data mining systems. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 432–439. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45372-5_47

5. Liu, B., Hsu, W.: Post-analysis of learned rules. In: AAAI/IAAI, vol. 1, pp. 828–834 (1996)
6. Liu, B., Hsu, W., Ma, Y.: Pruning and summarizing the discovered associations. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 125–134. ACM (1999)
7. Cao, L., Zhang, C.: Domain-driven, actionable knowledge discovery. IEEE Intell. Syst. **22**(4) (2007)
8. Cao, L., Zhao, Y., Zhang, H., Luo, D., Zhang, C., Park, E.K.: Flexible frameworks for actionable knowledge discovery. IEEE Trans. Knowl. Data Eng. **22**(9), 1299–1312 (2010)
9. Du, J., Hu, Y., Ling, C.X., Fan, M., Liu, M.: Efficient action extraction with many-to-many relationship between actions and features. In: van Ditmarsch, H., Lang, J., Ju, S. (eds.) LORI 2011. LNCS (LNAI), vol. 6953, pp. 384–385. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24130-7_29
10. Karim, M., Rahman, R.M.: Decision tree and naive bayes algorithm for classification and generation of actionable knowledge for direct marketing. J. Softw. Eng. Appl. **6**(04), 196 (2013)
11. Yang, Q., Yin, J., Ling, C., Pan, R.: Extracting actionable knowledge from decision trees. IEEE Trans. Knowl. Data Eng. **19**(1), 43–56 (2007)
12. Yang, Q., Yin, J., Ling, C.X., Chen, T.: Postprocessing decision trees to extract actionable knowledge. In: Third IEEE International Conference on Data Mining, ICDM 2003, pp. 685–688. IEEE (2003)
13. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning. Springer Series in Statistics, vol. 1. Springer, New York (2001). https://doi.org/10.1007/978-0-387-21606-5
14. Cui, Z., Chen, W., He, Y., Chen, Y.: Optimal action extraction for random forests and boosted trees. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 179–188. ACM (2015)
15. Manindra, A., Thomas, T.: Satisfiability problems. Technical report (2000)
16. CPLEX, I.I.: V12. 1: User's manual for cplex. Int. Bus. Mach. Corp. **46**(53), 157 (2009)
17. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002)
18. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, pp. 785–794. ACM (2016)
19. Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., Moore, R.: Real-time human pose recognition in parts from single depth images. Commun. ACM **56**(1), 116–124 (2013)
20. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
21. Tyree, S., Weinberger, K.Q., Agrawal, K., Paykin, J.: Parallel boosted regression trees for web search ranking. In: Proceedings of the 20th International Conference on World Wide Web, pp. 387–396. ACM (2011)