NAIST-IS-MT1651069

Master's Thesis

Beyond the Real: Alternative Methods for Discovering Translation Enhancer via Machine Learning

Hiroaki Tanaka

March 1, 2018

Graduate School of Information Science
Nara Institute of Science and Technology

A Master's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Master of ENGINEERING

Hiroaki Tanaka

Thesis Committee:

Professor Satoshi Nakamura	(Supervisor)
Professor Shigehiko Kanaya	(Co-supervisor)
Associate Professor Yu Suzuki	(Co-supervisor)
Associate Professor Ko Kato	(Co-supervisor)
Assistant Professor Koichiro Yoshino	(Co-supervisor)

Beyond the Real: Alternative Methods for Discovering Translation Enhancer via Machine Learning *

Hiroaki Tanaka

Abstract

Protein production by plants is a hot topic, because the plants have some advantages in the application for drug discovery compared to other hosts, e.g., bacteria, yeasts, animals, etc. HoIver, it is a problem that the amount of protein expression in plants is less than that of the other hosts. Many researchers tackle this problem and some researchers succeed to increase the amount of produced proteins by changing original 5'UTR to artificial 5'UTR; 5'UTR is one of the areas of mRNA (messenger RNA) and affect the translation from mRNA to proteins. Albeit the important sequence, discovering translation enhancer referring to the 5'UTR sequences which increase the amount of translated proteins is difficult, because the experiments in discovering the sequence require the significant cost, time and effort. To solve this problem, I want to predict how a large amount of translated proteins are obtained from given mRNAs without the experiments. In this paper, I propose a method R-STEINER (generate nucleotide sequence Randomly and Select a TrEmendous 5'-untranslated region which INcrEase the amount of tRanslated proteins of a certain gene). In R-STEINER, I build a model predicting the amount of translated proteins—this prediction model of translated proteins from mRNA sequence is the world's first model—, generates 5'UTR sequences and discovers the 5'UTRs which will increase the amount of translated proteins by using the prediction model. Using this method, I can obtain the translation enhancers without real synthesis experiments, leading to reduction of the cost, time and effort of the experiments. In my study, I built the prediction model by *Oryza sativa* (rice) and synthesized the 5'UTRs which Ire generated by R-STEINER. This study confirmed that the model can predict the amount of translated proteins with a correlation coefficient is 0.89.

*Master's Thesis, Graduate School of Information Science,
Nara Institute of Science and Technology, NAIST-IS-MT1651069, March 1, 2018.

Some companies own patents of 5'UTR sequences, and they require a technique to make new 5'UTR sequences which increase the amount of translated proteins based on their patent sequences. In this paper, I tackled solving this requirement. For solving the requirement, I have to develop a technique to obtain any input vectors that satisfy a condition to change, increase or decrease, the objective variable. In this paper, I propose a method, named TRANS-AM (TRANSforming-feAture Method), which can discover an input vector satisfying the condition of objective variable in regression problems and in the case of using random forest regression, by using a property of regression tree. The regression tree splits an input space into subspaces. There are subspaces with corresponding to objective variables satisfying the condition—the corresponding objective variables are higher than a given threshold. By transforming the input vector to new input vectors belonging to one of the subspaces, I can discover a new input vector whose distance from the original input vector is minimum by satisfying the condition to change the objective variable. I evaluated the proposal method through numerical simulations and investigated that the proposal method worked Ill for the datasets generated through several processes. In the final part of this paper, I suggest the application of TRANS-AM for 5'UTR generation.

Keywords:

bioinformatics, machine learning, protein production, random forest, gradient boosting, XGBoost, mRNA, protein synthesis, translation efficiency, gene expression, feature extraction, actionable knowledge extraction

Contents

List of Figures

List of Tables

Acknowledgement

I thank all of the relevant members for support. First, I would like to thank Professor Satoshi Nakamura so much for giving me a lot of useful advice and for organizing the research environment. I also would like to thank Professor Kanaya Shigehiko for giving me a lot of brilliant advice. Associate Professor Yu Suzuki supported me in many situations: writing papers, attending conferences, discussion for study and developing a comfortable research environment. I'm deeply indebted to him. I appreciate Associate Professor Ko Kato and Dr. Shotaro Yamasaki in department of biological science providing us the wonderful dataset, teaching me the knowledge of bioscience and discussing intensely. Assistant Professor Koichiro Yoshino also gave me helpful suggestions about the research and helped write the papers. I owe a deep debt of a gratitude to him.

I'm grateful for Ms.Miho Hayashi and Ms.Manami Matsuda. They always supported and encouraged me so that I can work comfortably. I also appreciate Ms.Shiori Yamaguchi correcting my clumsy English of this paper. Finally, I'm full of gratitude for the members of Augmented Human Communication Laboratory and my family. They always gave me wonderful wonderful daily life.

Thank you, all of the relevant members.

Chapter 1

Introduction

Protein production in plants has been one of the hot topics (Maxmen et al., 2012). I have several options for the platform used for protein production and each platform has advantages and disadvantages. One example is about drug development, which should first take priority over the safety. If I develop the vaccines by using human cells, these vaccines potentially bring serious harm to human bodies. On the other hand, if I use plants or plants' cells, the risk can be avoided (Yao, Weng, Dickey, & Wang, 2015). Although using plants have many benefits, the significant disadvantage exists: the amount of produced proteins in plants are less than that of the other hosts (e.g. animal, bacteria, yeast). Therefore, the techniques which increase the amount of produced proteins are required.

The process that produces proteins from DNA is divided into two parts, transcription and translation. Not so many studies regarding translation have been carried out relative to the studies focusing on transcription. HoIver, the amount of produced proteins are decided by these two parts, i.e., the translation is an important factor to increase the amount of produced proteins. Therefore, in this paper, I focus on the translation part and I aim to increase the amount of translated proteins.

Specifically, my study focuses on the relationship betIen mRNA (messenger RNA) and the amount of translated proteins. The 5'UTR (5'-translated region) affects the amount of translated proteins, and Yamasaki et al. (2018) succeeded to increase the amount of translated proteins of a certain gene by editing 5'UTR sequences. HoIver, the real experiments to discover the translation enhancers—referring to the 5'UTR sequences which increase the amount of translated proteins of a certain gene— require the significant cost, time and effort. In this paper, I will propose a method, named R-STEINER (generate nucleotide sequence Randomly and Select a TrEmendous 5'-

untranslated region which INcrEase the amount of tRanslated proteins of a certain gene). By using R-STEINER, I can obtain the translation enhancers without real experiments, leading to reduction of the cost. The details of this method are explained in Chapter 3.

I will take another issue of protein production. Some companies have already taken the patent related to 5'UTR sequences, and these companies and researchers have been trying to improve the amount of translated proteins by utilizing their 5'UTR sequences. HoIver, my first method generates new 5'UTR sequences, i.e., the method cannot utilize their sequences. To solve this issue, I will propose a new method, named TRANS-AM (TRANSforming-feAture Method). This method is useful not only in generating 5'UTR sequences but also in other regression tasks. Details on TRANS-AM are given in Chapter 4.

This paper is composed of the following parts. First, in Chapter 2 I introduce the basic knowledge of bioscience and machine learning. Understanding the knowledge may be complex, but it is unnecessary to know the details of these topics in order to understand this paper. Then in Chapter 3, I propose the first method, R-STEINER, folloId by introducing the second method, TRANS-AM in Chapter 4. Chapter 3 and Chapter 4 are independent with each other. Finally, in Chapter 5, I give the conclusion in this paper and the discussion for future work. In this chapter, I argue that the TRANS-AM should be useful to solve the industrial requirement.

Chapter 2

Basic Knowledge

In this chapter, I introduce basic knowledge to in understanding my study.

2.1 Translation

I show the process from the gene to proteins in Figure 2.1. "Translation" indicates the process from mRNA to proteins. The mRNA consists of three areas: 5'UTR, CDS (coding DNA sequence) and 3'UTR (3'-untranslated region) (see Figure 2.2). Then the mRNA is decoded in a ribosome to produce a specific amino acid chain, or protein. Strictly, only some combinations in CDS are decoded to amino acid, for example the combination GCU is decoded to $C_3H_7NO_2$: alanine (see Figure 2.2). More details of the translation is explained in Alberts et al. (2013).

In my study, I aim to make a large amount of proteins from one mRNA. If any mRNA is decoded to protein actively, many ribosomes are attached to the mRNA, and the ribosomes and mRNA form a complex called polysome. In other words, as the ratio of mRNA which forms the polysome increases, the amount of proteins generated from that mRNA increase. Therefore I use the ratio of mRNA as the criterion of the amount of translated proteins, i.e., how much proteins are generated from the mRNA.

Figure 2.1: process from the gene to proteins

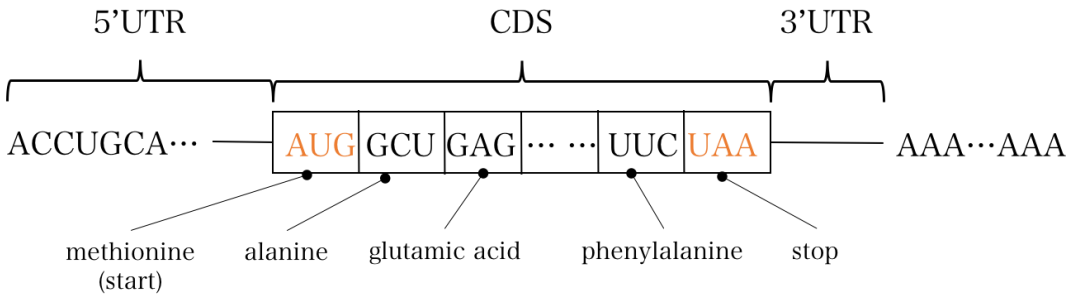

Figure 2.2: structure of mRNA

I call the ratio as PR-value (Polysome Ratio value).

It is known that an area of 5'UTR affects the amount of translated proteins (Sugio, Satoh, Matsuura, Shinmyo, & Kato, 2008). According to this report, I have assumed to be able to improve the amount of translated proteins by controlling the sequence of 5'UTR.

I have two reasons why I control only the 5'UTR sequence. First, controlling CDS is not reasonable for applications. If I change the CDS sequence, the proteins produced by mRNA are changed. This is not desirable for applications. Second, controlling the 3'UTR sequence is almost impossible. The 3'UTR contains information on where the 3'UTR will break. Therefore, if I try to control the 3'UTR sequence, the synthesized 3'UTR may be unexpected sequence. According to above two reasons, I do not change the CDS sequence and 3'UTR sequence.

I introduce one more basic knowledge of the mRNA. The mRNA usually makes a secondary structure, i.e, the mRNA does not lie on a straight line but make a complex structure (see Figure 2.3). I can estimate the possible forms and free energy of it from the nucleotide sequence. The free energy indicate how strongly the nucleotide sequence makes the secondary structure—as the nucleotide sequence makes the secondary structure more strongly, the free energy increase. In this study, I calculated the free energy by ViennaRNA Package (Lorenz et al., 2011).

2.2 Regression Model

Suppose that I have the dataset such as $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)$, where $\boldsymbol{x}_n \in \mathcal{X} \subset \mathbb{R}^d, y_n \in \mathcal{Y} \subset \mathbb{R}$ and that there is a relationship between $\boldsymbol{x}_n$ and y_n such as $f : \mathcal{X} \to \mathcal{Y}$.

Regression is to estimate the map f, and I call the estimated f as a regression

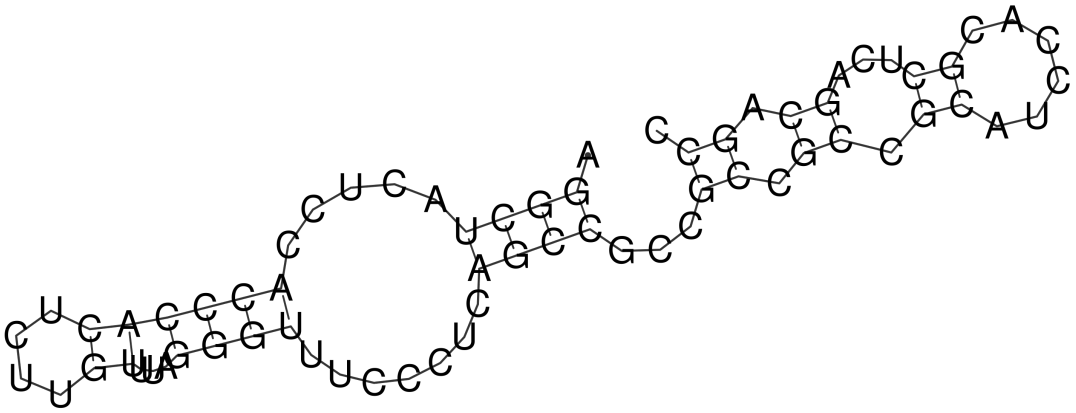

Figure 2.3: example of secondary structure

model, prediction model, model or predictor. I introduce six models to estimate f. Generally, the method for estimating f consists of two parts: (1) define a class [1] $\mathcal{H}$ of function f, (2) pick up one function $\hat{f}$ from $\mathcal{H}$ which can estimate the most accurate f. In other words, (1) is equal to restricting the form of f, and (2) is equal to determining some parameters contained in f by optimization of a loss function ℓ. The ℓ measures the error between predicted and observed values.

Then I can obtain a prediction model $\hat{f}$, and for some new data $\boldsymbol{x}^*$ I can predict y^* by $\hat{f}(\boldsymbol{x}^*)$.

2.2.1 Linear Regression

Linear regression is the most fundamental method to estimate f by the following formula

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} b_i x_i + b_0, \tag{2.1}$$

where x_i is the component of $\boldsymbol{x}$ and b_i is the estimated parameter. Defining the form of f as Equation (2.1) corresponds to define the class $\mathcal{H}$ as

$$\mathcal{H} = \left\{ \sum_{i=1}^{d} b_i x_i + b_0 \;\middle|\; b_i \in \mathbb{R} \right\}. \tag{2.2}$$

[1] A class is just like the set of functions. For example, the class $\{\, ax + b \mid a, b : \text{const.} \,\}$ contains x, $x + 1$, $2x + 1$, etc...

Then I pick up a function for estimating f, i.e., I determine the parameters by solving an optimization problem

$$\hat{\boldsymbol{b}} = \underset{\boldsymbol{b}\in\mathbb{R}^{d+1}}{\arg\min} \frac{1}{N}\sum_{n=1}^{N} \ell(\boldsymbol{x}_n, y_n) = \underset{\boldsymbol{b}\in\mathbb{R}^{d+1}}{\arg\min} \frac{1}{N}\sum_{n=1}^{N} \left(y_n - f(\boldsymbol{x}_n)\right)^2, \tag{2.3}$$

where $\boldsymbol{b}$ is the parameter vector containing $b_0, b_1, \cdots, b_d$.

Finally, I obtain the following prediction model

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{d} \hat{b}_i x_i + \hat{b}_0. \tag{2.4}$$

The solution of Equation (2.3) is written in J. Friedman, Hastie, and Tibshirani (2001), and more theoretical background of regression analysis is explained in detail by Seber and Lee (2012); Rao, Rao, Statistiker, Rao, and Rao (1973).

2.2.2 Lasso

Lasso (Least absolute shrinkage and selection operator) is a method to estimate f by Equation (2.1). In the lasso, $\boldsymbol{b}$ is estimated by

$$\hat{\boldsymbol{b}}_{\text{lasso}} = \underset{\boldsymbol{b}}{\arg\min} \left[\frac{1}{N}\sum_{n=1}^{N} \left(y_n - f(\boldsymbol{x}_n)\right)^2 + \lambda \sum_{i=0}^{d} |b_i| \right]. \tag{2.5}$$

In general, the parameter vector estimated by lasso contains many 0 components. If $\boldsymbol{x}$ has many components, i.e., if the feature vector has many features, the lasso can select important features automatically. The details of the lasso is explained in Tibshirani (1996); J. Friedman, Hastie, Höfling, Tibshirani, et al. (2007); J. Friedman, Hastie, and Tibshirani (2010).

2.2.3 PLS Regression

PLS (Partial Least Squares) regression is a method to estimate f. The critical point of the PLS regression is the method to estimate $\boldsymbol{b}$. In the PLS regression, $\boldsymbol{b}$ is estimated in a new projected space. The new space is structured such as the covariance betIen projected points of $(\boldsymbol{x}_n, y_n)$ are maximized. J. Friedman et al. (2001) claims that the PLS regression is likely to be similar to Ridge regression (Hoerl & Kennard, 1970) and principal component regression. The algorithm and the details of the PLS regression is explained in H. Wold (1985); S. Wold, Sjöström, and Eriksson (2001).

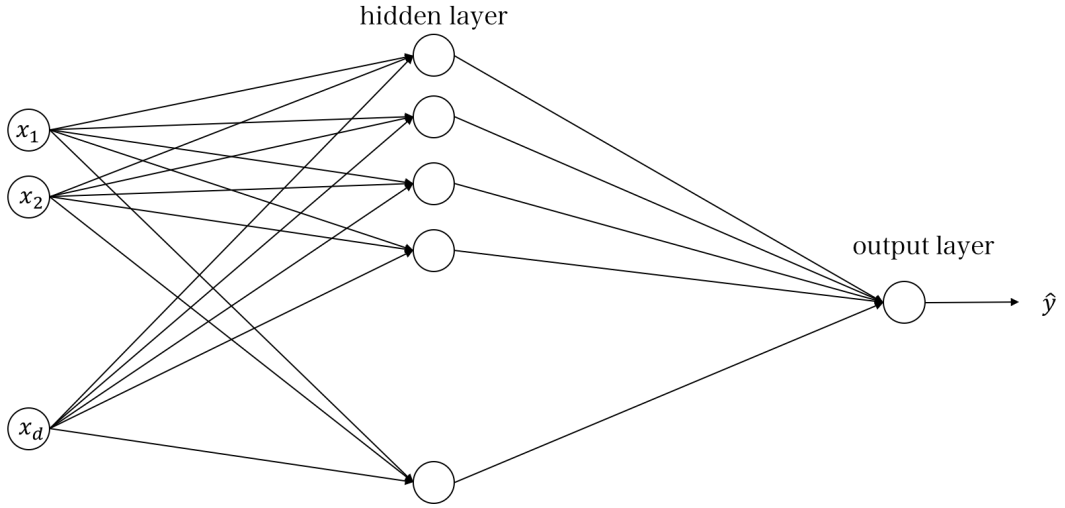

Figure 2.4: Example of Neural Network

2.2.4 Neural Network

A neural network (McCulloch & Pitts, 1943) is also a method to estimate f. The neural network has some layers and each layer has some units. In this paper, I call the first layer as an input layer, the final layer as an output layer and each middle layer between them as hidden layers.

Let us consider a simple example of the neural network containing one hidden layer and one unit in the output layer (see Figure 2.4). The j-th unit in the hidden layer receives $x_1, x_2, \cdots, x_d$ and yields

$$z_j = u_j \left(\sum_{i=1}^{d} b_{ji} x_i + b_j \right), \tag{2.6}$$

where u_j is called activation function and should be selected in advance. Then this neural network yields

$$\hat{y} = u^{(\text{out})} \left(\sum_{j=1}^{J} b_j z_j + b^{(\text{out})} \right) \tag{2.7}$$

I can use various activation functions. The classical activation function is a logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.8}$$

or tanh. Nowadays, instead of these classical functions, rectified linear function

$$\sigma(x) = \max\{x, 0\} \tag{2.9}$$

is adopted as the activation function.

For generalization, I introduce the matrix notation of Equation (2.6). Equation (2.6) is rewritten by

$$\boldsymbol{z} = u\left(B\boldsymbol{x} + \boldsymbol{b}_0\right), \tag{2.10}$$

where

$$\boldsymbol{z} = \begin{bmatrix} z_1 & \cdots & z_J \end{bmatrix}^T, \quad \boldsymbol{b}_0 = \begin{bmatrix} b_1 & \cdots & b_J \end{bmatrix}^T, \quad \boldsymbol{z} = \begin{bmatrix} z_1 & \cdots & z_J \end{bmatrix}^T,$$
$$B = \begin{bmatrix} b_{11} & \cdots & b_{1d} \\ \vdots & \ddots & \vdots \\ b_{J1} & \cdots & b_{Jd} \end{bmatrix}, \quad u(\boldsymbol{x}) = \begin{bmatrix} u(x_1) & \cdots & u(x_d) \end{bmatrix}^T.$$

According to above notations, I can write the output of any layer l by

$$\boldsymbol{z}^{(l)} = u\left(B^{(l)}\boldsymbol{z}^{(l-1)} + \boldsymbol{b}_0^{(l)}\right) \tag{2.11}$$

Lately, the multi-layer neural network has been very popular in various domains (Vaswani et al., 2017; Shazeer et al., 2017; Gehring, Auli, Grangier, Yarats, & Dauphin, 2017; Ye et al., 2017; Tjandra, Sakti, & Nakamura, 2017). HoIver, in the prediction of the amount of translated protein from mRNA, I cannot find the previous research using any multi-layer neural network.

2.2.5 Tree Based Ensemble Methods

I will introduce three predictors: random forest (Breiman, 2001), gradient boosting (J. H. Friedman, 2002) and XGBoost (Chen & Guestrin, 2016). These machine learning models make an important rule in my study.

As a basic knowledge to understand these predictors, firstly I introduce a regression tree. The regression tree divides the input space $\mathcal{X}$ into subspaces as

$$\mathcal{X} = \mathcal{X}_1 \oplus \mathcal{X}_2 \oplus \mathcal{X}_M, \tag{2.12}$$

and γ_m corresponds to the area $\mathcal{X}_m$. Then the regression tree estimates the objective variable y by

$$\hat{f}(\boldsymbol{x}) = \sum_{m=1}^{M} \gamma_m 1\left[\boldsymbol{x} \in \mathcal{X}_m\right], \tag{2.13}$$

where $1[\boldsymbol{x} \in \mathcal{X}_m]$ denotes an indicator function which is defined as

$$1\left[\boldsymbol{x} \in \mathcal{X}_m\right] = \begin{cases} 1 & \boldsymbol{x} \in \mathcal{X}_m \\ 0 & \boldsymbol{x} \notin \mathcal{X}_m \end{cases}. \tag{2.14}$$

I can rewrite Equation (2.12) by

$$\hat{f}(\boldsymbol{x}) = \begin{cases} \gamma_1 & \boldsymbol{x} \in \mathcal{X}_1 \\ \gamma_2 & \boldsymbol{x} \in \mathcal{X}_2 \\ \vdots & \\ \gamma_M & \boldsymbol{x} \in \mathcal{X}_M \end{cases} . \tag{2.15}$$

Random forest is an aggregation of some regression trees. In the random forest, I build some regression trees and estimate the objective variable y by the sample mean of the regression trees. When I build the regression trees, I use subsets of samples, i.e., I select some samples from the original dataset (This selection is called resampling).

Gradient boosting and XGBoost are also ensembles of some regression trees. In these two methods, at the resampling, I arbitrarily select some samples and estimate y by the weighted sample mean. The difference betIen the gradient boosting and the XGBoost is the algorithm in deciding these weights.

J. Friedman et al. (2001) stated that additive tree models have been widely applied to many industrial products such as Kinect (Shotton et al., 2013) and face detection in cameras (Viola & Jones, 2004), and these models should be attempted first for many data mining competitions such as Ib search ranking (Tyree, Weinberger, Agrawal, & Paykin, 2011).

Chapter 3

R-STEINER

In this chapter, I propose a first method, R-STEINER for 5'UTR sequence generation.

3.1 Introduction

As I described in Chapter 1, there are various advantages of the protein production by plants. Generally protein expression of the plant is less than that of other hosts (Fujiyama, n.d.; Yamasaki, Ueda, & Kato, 2013; Yao et al., 2015). As a solution, I focus on the area 5'UTR. It is known that gene expression is affected by sequence of 5'UTR (Yamasaki et al., 2018), and some researchers have tried to discover 5'UTR sequences that increase the amount of translated proteins of certain genes. However, real synthesis experiments done to discover such 5'UTR sequences involve significant costs and require time and effort.

As a solution, I propose a method, R-STEINER, that enable me to obtain 5'UTR sequences that increase the amount of translated proteins of a given gene without real synthesis experiments, resulting in reduced cost, time and effort. The proposed process is composed of the following parts: building a model for predicting the amount of translated proteins, generating 5'UTR sequences randomly and selecting those that increase the amount of proteins according to the model. The parts of sequence generation and selection involve performing a synthesis experiment; conventionally, I could obtain the amount of translated proteins only in real experiments, but the model gives me the predicted amount of proteins.

With R-STEINER, I build a model for predicting the amount of translated proteins. I have to evaluate the model not only in the traditional way of evaluating a machine learning model but also through real synthesis experiments because the 5'UTR se-

quences generated in the generation part are completely artificial. As the model built in R-STEINER is learned by natural sequences, there is the possibility that it cannot predict the amount of translated proteins from artificial sequences. In addition to this concern, I am concerned that it might not be possible synthesize some artificial sequences; presumably, 5'UTR sequences are made by some unknown rule, but sequences generated by R-STEINER are not. Given the above two concerns, I have to evaluate whether the model can predict the amount of translated proteins even for artificial 5'UTRs. For this evaluation, I performed real synthesis experiments and evaluated the model in Section 3.5.

3.2 Related Work

In this section, I introduce related work on the relationships among features of the mRNA sequence and translation efficiency.

Kawaguchi and Bailey-Serres (2005) analysed the relationships between ribosome loading[1], three features, the length of 5'UTR, CDS and 3'UTR, and the contents of A, U, G, C, AU, GC, CU, AG, GU and AC, respectively. Ribosome loading represents the translation efficiency. However, they analysed the relationships between only one feature and the translation efficiency. Therefore, they could not reveal the relationships among the translation efficiency and some features.

Matsuura et al. (2013) built a model for predicting relative F-Luc activity that uses PLS regression. The relative F-Luc activity represents how strongly heat-stress conditions affect the translation efficiency. The PLS regression model can take into account the relationships between some features; therefore, the problem that remains by Kawaguchi and Bailey-Serres (2005) is solved. However, the prediction precision was not sufficient in the case of predicting the PR-value (Section 3.4).

3.3 R-STEINER: Proposed Method

In this section, I propose my method, R-STEINER (generate nucleotide sequences Randomly and Select a TrEmendous 5'-untranslated region that Increases the traNSlation efficiency of a certain gene) to find 5'UTRs that allow certain genes to increase the amount of translated proteins. R-STEINER is split into two steps: the B-step where I build a model for predicting PR-value and the G-step where I generate 5'UTRs fol-

[1]Ribosome loading is one criteria of translation efficiency.

Table 3.1: Summary of *Con* Dataset ($N_{\mathrm{Con}} = 24915$)

Gene ID	5'UTR	CDS	3'UTR	PR-value
1	GUU...GAG	AUGU...AUGA	UGA...UGC	0.9229
2	GAA...UAU	AUGA...GUAA	GAG...GUC	1.0054
⋮	⋮	⋮	⋮	⋮
N_{Con}	$s^{5\mathrm{utr}}_{N_{\mathrm{Con}}}$	$s^{\mathrm{cds}}_{N_{\mathrm{Con}}}$	$s^{3\mathrm{utr}}_{N_{\mathrm{Con}}}$	$y_{N_{\mathrm{Con}}}$

lowed by selecting top-k sequences that increase the amount of translated proteins of a given gene. Details on B-step and G-step are given in Section 3.3.1 and Section 3.3.2, respectively.

3.3.1 B-step

The B-step consists of two steps;:

(B1) feature engineering;

(B2) building the prediction model;

In step (B1), I transform an mRNA sequence to a feature vector that I can throw into machine learning models. In step (B2), I build a model for predicting the PR-value by using an ensemble of random forest, gradient boosting and XGBoost.

I have two datasets that are in two conditions. The first dataset is for a normal condition. In this condition, the cells proliferate activity, and their matter production is active. I represent this dataset as *Con*. The second dataset is for a heat-stress condition. In this condition, the activities of cells are restrained, and, in general, their matter production is also reduced. I represent this dataset as *HS*. Summaries of the datasets for my analysis are shown in Tab. 3.1 and Tab. 3.2. Generally, the lengths of each area are different, i.e., the length of 5'UTR of gene n is different from that of gene $n + 1$. I split the datasets into a training set (50%), validation set (25%) and test set (25%). I trained predictors with the training set and tuned hyperparameters with the validation set. Then, in the aggregation step, I aggregate the three models, random forest, gradient boosting and XGBoost, because there is no difference among the prediction precision of these models.

(B1) Feature Engineering

In (B1), I engineer features for regression models. I use three types of features:

Table 3.2: Summary of *HS* Dataset ($N_{\mathrm{HS}} = 21786$)

Gene ID	5'UTR	CDS	3'UTR	PR-value
1	GAA...UAU	AUGA...GUAA	GAG...GUC	1.1293
2	AGG...GCC	AUGG...UUGA	GUG...UUC	0.7600
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N_{HS}	$s^{\mathrm{5utr}}_{N_{\mathrm{HS}}}$	$s^{\mathrm{cds}}_{N_{\mathrm{HS}}}$	$s^{\mathrm{3utr}}_{N_{\mathrm{HS}}}$	$y_{N_{\mathrm{HS}}}$

(F1) lengths of 5'UTR, CDS and 3'UTR,

(F2) secondary energies of 5'UTR, CDS and 3'UTR,

(F3) counts of 3-gram acids of 5'UTR, CDS and 3'UTR.

It is known that (F1) and (F2) affect the amount of translated proteins (Kawaguchi & Bailey-Serres, 2005) under experimental settings. Using these features, I construct the following feature vector:

$$\boldsymbol{x} = \mathrm{concat}\left[\boldsymbol{x}_{F_1} \quad \boldsymbol{x}_{F_2} \quad \boldsymbol{x}_{F_3}\right] \in \mathbb{R}^{238}, \tag{3.1}$$

where

$$\boldsymbol{x}_{F_1} = \left[\mathrm{len}(5'\mathrm{UTR}) \quad \mathrm{len}(\mathrm{CDS}) \quad \mathrm{len}(3'\mathrm{UTR})\right] \in \mathbb{R}^3, \tag{3.2}$$

$$\boldsymbol{x}_{F_2} = \left[G(5'\mathrm{UTR}) \quad G(\mathrm{CDS}) \quad G(3'\mathrm{UTR})\right] \in \mathbb{R}^3, \tag{3.3}$$

$$\boldsymbol{x}_{F_3} = \mathrm{concat}\left[\boldsymbol{c}^{5'\mathrm{UTR}} \quad \boldsymbol{c}^{\mathrm{CDS}} \quad \boldsymbol{c}^{3'\mathrm{UTR}}\right] \in \mathbb{R}^{232}. \tag{3.4}$$

Here, I use following notations. $\mathrm{len}(R)$ and $G(R)$ represent the length of any area R and the free energy of R, respectively. $\boldsymbol{c}^{5'\mathrm{UTR}}$ and $\boldsymbol{c}^{3'\mathrm{UTR}}$ contain counters of A, U, G, C, AA, AU, ..., UU, AAA, AAU, AAU, ..., UUU on 5'UTR and 3'UTR, respectively. $\boldsymbol{c}^{\mathrm{CDS}}$ represents AAA, AAU, ..., UUU on CDS.

In CDS, three continuous nucleotides correspond to an amino acid; therefore, I assumed that the counters of two continuous nucleotides and one nucleotide do not need to be counted. Kawaguchi and Bailey-Serres (2005) also analysed the relationships between A, U, G and C contents and the translation efficiency. In my study, I use more various count features than the features used by Kawaguchi and Bailey-Serres.

(B2) Prediction Model

I build a model predicting the PR-value by using an ensemble model of six models comprising random forest models, gradient boosting models and XGBoost models, i.e.,

I estimate the PR-value of a given mRNA by using Equation (3.7).

$$\hat{h}^{(\mathrm{HS})}\left(\boldsymbol{x}^{*}\right)=\frac{1}{3}\left(h_{\mathrm{rf}}^{(\mathrm{HS})}\left(\boldsymbol{x}^{*}\right)+h_{\mathrm{gb}}^{(\mathrm{HS})}\left(\boldsymbol{x}^{*}\right)+h_{\mathrm{xgb}}^{(\mathrm{HS})}\left(\boldsymbol{x}^{*}\right)\right), \tag{3.5}$$

$$\hat{h}^{(\mathrm{Con})}\left(\boldsymbol{x}^{*}\right)=\frac{1}{3}\left(h_{\mathrm{rf}}^{(\mathrm{Con})}\left(\boldsymbol{x}^{*}\right)+h_{\mathrm{gb}}^{(\mathrm{Con})}\left(\boldsymbol{x}^{*}\right)+h_{\mathrm{xgb}}^{(\mathrm{Con})}\left(\boldsymbol{x}^{*}\right)\right), \tag{3.6}$$

$$\hat{h}\left(\boldsymbol{x}^{*}\right)=\frac{1}{2}\left(\hat{h}^{(\mathrm{HS})}\left(\boldsymbol{x}^{*}\right)+\hat{h}^{(\mathrm{Con})}\left(\boldsymbol{x}^{*}\right)\right), \tag{3.7}$$

where $\boldsymbol{x}^{*}$ is the feature vector of the given mRNA, $h_{\mathrm{rf}}^{(\mathrm{HS})}(\cdot), h_{\mathrm{gb}}^{(\mathrm{HS})}(\cdot), h_{\mathrm{xgb}}^{(\mathrm{HS})}(\cdot)$ mean the prediction model built by random forest, gradient boosting and XGBoost in *HS*, respectively and $h_{\mathrm{rf}}^{(\mathrm{Con})}(\cdot), h_{\mathrm{gb}}^{(\mathrm{Con})}(\cdot), h_{\mathrm{xgb}}^{(\mathrm{Con})}(\cdot)$ also mean the prediction models in *Con* vice versa.

3.3.2 G-step

The G-step is also split into three steps: random generation, selecting good feature vectors, and making sequences. In the random-generation step, I generate nucleotides (A, U, G or C) randomly and then obtain sequences on a computer. In the step for selecting good feature vectors, I transform the sequences obtained in the previous step into the feature vector and predict the PR-value by using the ensemble prediction model comprising the three prediction models. Then, I select the top k feature vectors whose PR-values are largest. In the step of making a sequence, I make sequences corresponding to the selected feature vector. Here, I should pay attention to the relationship between feature vector and sequence. Feature vectors do not correspond to sequences one to one. Therefore, I have to develop an algorithm to make a feature vector correspond to a sequence.

Sequence Generation

In the G-step, I generate B 5'UTR sequences and combine them with certain CDS and 3'UTR. Then, for the combined mRNA, I predict the PR-value by using the prediction model and select the top k mRNA sequences. In the generation of 5'UTR, I generate ℓ nucleotides (A, U, G, C) randomly and make mRNA by combining the nucleotides. I cannot use the sub-sequences AUG and AAUAAU. Thus, I generate 5'UTR by using Algorithm 1. In this paper, I use $B = 2 \times 10^{6}$.

Algorithm 1 Algorithm of mRNA Generation

Require: $\hat{h}(\cdot)$: prediction model defined by Equation (3.7)

Ensure: k sequences of 5'UTR $s^{5'\mathrm{UTR}}$

1: make one part of feature vectors $\boldsymbol{x}_{F_2}$ and $\boldsymbol{x}_{F3}$
2: **for** $t = 1, 2, \cdots, B$ **do**
3: fix $L \in \mathbb{N}$ randomly in interval $(22, 49)$
4: generate acids $\{ s_\ell \}_{\ell=1}^{L}$, where s_ℓ is selected from $\{ \mathrm{A}, \mathrm{U}, \mathrm{G}, \mathrm{C} \}$ randomly
5: $S_t^{5'\mathrm{UTR}} \leftarrow \mathrm{concat} \{ s_\ell \}_{\ell=1}^{L}$
6: make one part of feature vector $\boldsymbol{x}_{F_3}$ of $S_t^{5'\mathrm{UTR}}$
7: make the feature vector $\boldsymbol{x}_t^*$ by concatination of $\boldsymbol{x}_{F_1}, \boldsymbol{x}_{F_2}$ and $\boldsymbol{x}_{F_3}$
8: estimate PR-value of the sequence $S_t^{5'\mathrm{UTR}}$ at current step by

$$\hat{y}_t = \hat{h}\left(\boldsymbol{x}_t^*\right)$$

9: **end for**
10: sort $\{ S_t^{5'\mathrm{UTR}} \}_{t=1}^{T}$ in descending order of $\{ \hat{y}_t \}_{t=1}^{T}$
11: **return** top k sequences of $\{ S_t^{5'\mathrm{UTR}} \}_{t=1}^{T}$

3.4 Preliminary Evaluation of Prediction Models

In this section, I compare models developed with random forest, gradient boosting and XGBoost with the models developed by linear regression, PLS regression (H. Wold, 1966), linear lasso (Tibshirani, 1996; J. Friedman et al., 2007, 2010) and neural network (McCulloch & Pitts, 1943). Then, I declare that the tree-based prediction models are better than the others. As the feature vector $\boldsymbol{x}$ contains discrete and continuous variables, tree-based models will show better performance than the other models (J. Friedman et al., 2001).

Hyperparameter Tuning

Here, I explain how to tune the hyperparameters of the models. Each prediction model has hyperparameters. The PLS regression model has the hyperparameter η, which represents the number of principal components. The feed-forward neural network has many hyperparameters for the architecture of the model, such as the number of layers, the number of units in each layer and the activation functions in each layer. In addition to these hyperparameters, I should decide the learning rate, the way of optimization, whether should I drop out some units and so on. Generally, such an architecture,

hyperparameters and various network tunings are determined on the basis of previous research of the same domain. However, I could not find such research; therefore, I used a simple feed-forward neural network as my prediction model. The hyperparameters of random forest, gradient boosting and XGBoost that I tuned were the number of regression trees M and the maximum depth of each regression tree $d_{\max}$.

These hyperparameters were determined by using the validation set. There are various methods for optimizing hyperparameters, such as grid search, random search (Rastrigin, 1963; Schumer & Steiglitz, 1968; Schrack & Choit, 1976) and Bayesian optimization (Močkus, 1975; Mockus, 1977, 2012). Generally, for some loss function that is not represented clearly, I cannot obtain a strictly optimal solution for minimizing the function. Therefore, I seek the hyperparameter in a given search area, i.e., even if I adopt any of the above methods, I have to determine the area where hyperparameters are searched. If the search area is out of focus, the selected hyperparameters are far from the true optimal hyperparameters. Specifically, at the worst, I was concerned that the selected hyperparameters will not even be a local minimum point along one axial direction. To avoid such an unfortunate situation, I sought hyperparameters with the following steps.

step 1. Set initial value of hyperparameters as $(\theta_1, \theta_2, \cdots, \theta_p) = \left(\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_p\right)$.

step 2. Update $\theta_1 = \hat{\theta}_1$ such that it satisfies

$$\hat{\theta}_1 = \underset{\theta_1 \in I_1}{\arg\min} \frac{1}{|\mathcal{D}_{\text{vali}}|} \sum_{\boldsymbol{x}_n \in \mathcal{D}_{\text{vali}}} l\left(h(\boldsymbol{x}_n), y_n\right), \tag{3.8}$$

where $\mathcal{D}_{\text{vali}}$ represents the validation set, l is the squared loss, the other hyperparameters contained in the loss function l are fixed and interval I_1 contains the local minimum point along the θ_1 axial direction.

step 3. Update $\theta_2 = \hat{\theta}_2, \cdots, \theta_p = \hat{\theta}_p$ in a similar manner to previous step.

step 4. Finally, determine $\theta_1, \theta_2, \cdots, \theta_p$ by grid search in area $\prod_{i=1}^{p}[\theta_i - \varepsilon_i, \theta_i + \varepsilon_i]$.

By following these steps, I can focus narrowly on the area that contains at least one (local) minimum point, i.e., I can avoid the situation where the searched area does not contain any (local) minimum points. Certainly, if the prediction model has only one hyperparameter, all you need to do is search for the optimal value along the θ_1 axial direction.

If the prediction models have hyperparameters, they are tuned in the previous steps, except for the neural network. The areas for grid search and determined values

in each prediction model are shown in Tab. 3.4. I adopted the Bayesian optimization to determine each hyperparameter, the number of units and dropout rates in each layer, because the combination pattern is too large to adopt the grid search in the neural network. The architecture of the neural network is described in Tab. 3.3.

Table 3.3: Hyperparameters of neural network

layer	hyperparameter	candidate area	selected (HS)	selected (Con)
input	activation	None	tanh	tanh
hidden	number of units	$\{256, 512, 1024, 2148\}$	2048	512
	drop out rate	$[0, 0.5]$	0	0.27
	activation function	None	relu	relu
hidden	number of units	$\{256, 512, 1024, 2148\}$	1024	512
	drop out rate	$[0, 0.5]$	0	0
	activation function	None	linear	linear
output	number of units	None	1	1

Table 3.4: Hyperparameters

Model	hyperparameter	searched area (HS)	determined value (HS)	searched area (Con)	determined value (Con)
PLS regression	η	$\{2, 3, \cdots, 221\}$	123	$\{2, 3, \cdots, 221\}$	90
linear lasso	α	$\{2^i \mid i = 1, 0, \cdots, -4\}$	2^{-4}	$\{2^i \mid i = 1, 0, \cdots, -4\}$	2^{-4}
random forest	M	$\{10, 20, \cdots, 100\}$	94	$\{270, 275, \cdots, 285\}$	285
	$d_{\max}$	$\{1, 6, \cdots, 31\}$	20	$\{11, 12, \cdots, 20\}$	16
gradient boosting	M	$\{35, 40, \cdots, 50\}$	50	$\{180, 185, \cdots, 195\}$	195
	$d_{\max}$	$\{9, 10, \cdots, 14\}$	9	$\{1, 2, \cdots, 10\}$	5
XGBoost	M	$\{2^i \mid i = 11, 12, \cdots, 15\}$	2^{11}	$\{974, 984, \cdots, 1074\}$	1054
	$d_{\max}$	$\{1, 6, \cdots, 46\}$	11	$\{1, 2, \cdots, 10\}$	3

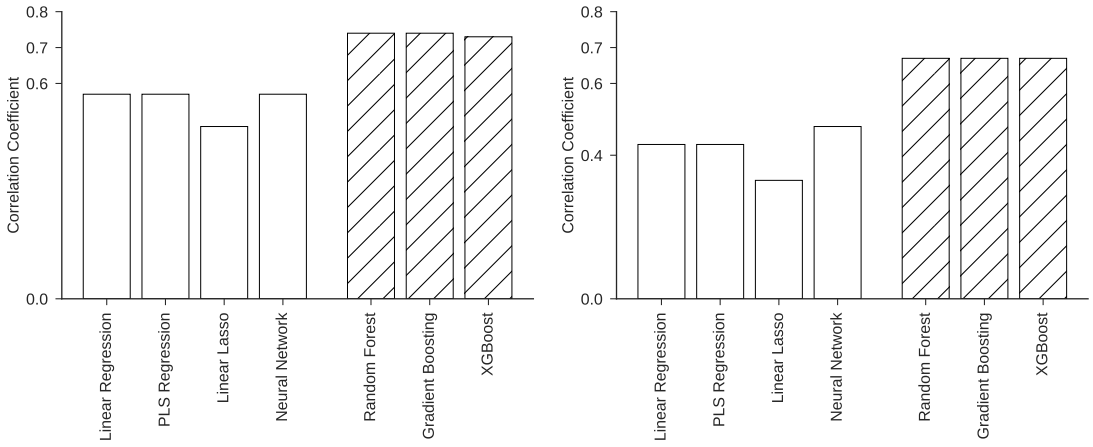

Figure 3.1: Correlation coefficients between observed and predicted value. Left is in *HS*; right is in *Con*.

Evaluation of Prediction Models

Here, I evaluate the prediction models by training the models with the training set, tuning the hyperparameters in the manner described in Section 3.4 with the validation set and comparing the models with the test set. The results are shown in Figure 3.1. As is shown, the tree-based prediction models were better than the other prediction models. In addition, I apply a statistical test:

Null Hypothesis H_0 $\rho_{\mathrm{rf}} = \rho_{\mathrm{gb}} = \rho_{\mathrm{xgb}}$,

Alternative Hypothesis H_1 $\neg H_0$,

where ρ_{rf}, ρ_{gb}, and ρ_{xgb} mean the correlation coefficients obtained by using random forest, gradient boosting and XGBoost, respectively. I cannot reject the null hypothesis, i.e., there is the potential for no differences in the correlation coefficients. Therefore, I aggregated all three models in Section 3.3.1.

3.5 Synthesis Experiment

In the G-step, I selected 5'UTR sequences by using the prediction model as the utility function, i.e., I selected the top-k 5'UTRs that maximized the predicted PR-value in the generated sequences. However, the prediction model is learned to fit the natural 5'UTRs, so the model does not always predict the PR-value of the artificial 5'UTRs accurately. Therefore, I had to conduct synthesis experiments in order to make sure

that the predicted PR-values of the artificial 5'UTRs were close to the true amounts of translated proteins. If the predicted PR-values are close to the true amounts of the proteins, I can obtain high-performance 5'UTRs by increasing the iterations of sequence generation in Algorithm 1.

In the experiments, I adopted the criterion F/R-luc activity, which represents the amount of translated proteins of the 5'UTR. It is known that $\log_{10}$(F/R-luc activity) bears a linear relationship with PR-value (Yamasaki, 2016). Hence, I evaluated whether the predicted PR-values were close to the observed translation efficiencies with the correlation coefficient. I chose 5'UTRs that were made in the experiments as follows.

(S1) Generate 5'UTR sequences as use R-STEINER, i.e., execute the algorithm until the 9-th line.

(S2) Select three 5'UTRs whose PR-values are highest, two 5'UTRs whose PR-values are lowest and four 5'UTRs randomly. Note that these four 5'UTRs are not same as previous three and two 5'UTRs.

I synthesized 5'UTRs that were selected in the above way and calculated the correlation coefficient between the predicted PR-value and observed $\log_{10}$(F/R-luc activity).

I clarify how the synthesis experiments were performed in 3.5.1 - 3.5.7. Then, I show the evaluation in 3.5.8.

3.5.1 Plant Materials, Culture Conditions, and Growth Conditions

Oryza sativa L. cv. Nipponbare suspension cells (Saotome et al., 2006) were cultured in R2S medium with rotary shaking at 90 rpm at 30°C in a dark condition. For genome-wide analysis of the polysome association, cells cultured for three days and cells cultured for three days and incubated at 41°C for 15 min were collected as the control (*Con*) sample and heat-stress (*HS*) sample, respectively. In addition, Oc suspension cells from roots of *Oryza sativa* L. accession C5924 (Baba, Hasezawa, & Syōno, 1986), that is, the suspension cells of the easy-to-isolate protoplast, were cultured under the same condition, and Oc cells cultured for three days were used for transient expression assay. Both suspension cell cultures were maintained with sub-culturing every week.

3.5.2 Polysome Fractionation Assays and RNA Isolation from Sucrose Gradients

Polysome fractionation analysis was performed according to the previously described method in Yamasaki, Matsuura, Demura, and Kato (2015). Cell extracts were layered

on a 26.25 - 71.25% sucrose density gradient and centrifuged. After centrifugation, the gradients were separated into two fractions by using a piston gradient fractionator (BioComp Instruments, Fredericton, NB, Canada). The second fraction of the bottom half (polysome fraction) and both fractions (total fraction) were individually collected and pooled into tubes containing guanidine hydrochloride (final concentration, 5.5 M). RNA was precipitated by the addition of an equal volume of ethanol, overnight incubation at $-20°C$ and centrifugation at $10,000$ rpm for 45 min in a JA-20 rotor (Beckman Coulter, Fullerton, CA, USA). The resulting precipitate was washed with 85% ethanol. RNA was purified by using an RNeasy kit (Qiagen, Hilden, Germany) with on-column DNase I treatment according to the manufacturer's instructions. RNA was eluted with 100 µl of RNase-free water, and the RNA integrity was examined with an Agilent Bioanalyzer 2100 (Agilent Technologies). I prepared RNA in two independent biological replicates.

3.5.3 Cap Analysis of Gene Expression (CAGE) and Data Analysis

nAnT-iCAGE libraries preparation, sequencing, filtering, mapping and gene annotation were performed on the basis of the previously described methods in Yamasaki et al. (2018). For this analysis, to more accurately identify the transcription start site (TSS), additional quality control was performed to remove tags with mismatches within three bases from the 5'end. In addition, tag counts were converted to tag per million (TPM) values at each TSS level as TPM_TSS and averaged between two replicates. Finally, I calculated the polysome ratio at each TSS level (PR_TSS) as an indicator of polysome association by using the following formula Equation (3.9). To obtain more reliable data, I used a limited TSS that mapped more than 50 tags in the total fraction data in both replicates for calculating PR_TSS. Genome information from IRGSP-1.0 in The Rice Annotation Project Database[2] was used as a reference for rRNA tag removal, mapping and annotation.

$$\text{PR\_TSS} = \frac{\text{TPM\_TSS in polysome fraction data}}{\text{TPM\_TSS in total fraction data}} \tag{3.9}$$

3.5.4 Plasmid Construction

A reporter plasmid for transient expression assay with PEG-mediated protoplast transformation using *in vitro* synthesized RNA was constructed by modifying plasmid pFLpA (Matsuura et al., 2013). The test 5'-UTRs were synthesized by oligonucleotide

[2]http://rapdb.dna.affrc.go.jp

annealing that included a part of the T3 promoter with NcoI site and a part of the F-luc coding region with the AatII site (Tab. 3.5). Each annealed oligonucleotide was introduced into pFL-pA at the NcoI/AatII sites to generate the plasmids pT3-5'-UTR-FL-pA. Insert DNA fragments were verified by sequencing. In Tab. 3.5, NcoI and AatII sites are shown in red and blue, respectively. Underlines represent T3 promoter regions.

Table 3.5: Gene-specific primers for *in vitro* synthesized RNA

5'UTR name	Direction	Primer sequence (5' to 3')
GS1	Forward	CATGGAATTAACCCTCACTAAAGGATTAATTGTACAGCACAGCAAATTGTCAACTATTTTTTCAGACAGATGGAAGACGT
	Reverse	CTTCCATCTGTCTGAAAAAATAGTTGACAATTTGCTGTGCTGTACAATTAATCCTTTAGTGAGGGTTAATTC
GS2	Forward	CATGGAATTAACCCTCACTAAAGGAAGTATCATTAAGTTTGATTTATTTGTCAAGGAACAAGTATCTTAGATGGAAGACGT
	Reverse	CTTCCATCTAAGATACTTGTTCCTTGACAAATAAATCAAACTTAATGATACTTCCTTTAGTGAGGGTTAATTC
GS3	Forward	CATGGAATTAACCCTCACTAAAGGTACAAACGAACTCTCATTCACATCATTAGATAAATTAATTTTAATATGGAAGACGT
	Reverse	CTTCCATATTAAAATTAATTTATCTAATGATGTGAATGAGAGTTCGTTTGTACCTTTAGTGAGGGTTAATTC
GS3	Forward	CATGGAATTAACCCTCACTAAAGGTTCCGGATCGTACCGGTCAGGATGGAAGACGT
	Reverse	CTTCCATCCTGACCGGTACGATCCGGAACCTTTAGTGAGGGTTAATTC
GS4	Forward	CATGGAATTAACCCTCACTAAAGGTACCAGCCCTAGGTGGAAACATTCTAGCCGCCATAATGGAAGACGT
	Reverse	CTTCCATTATGGCGGCTAGAATGTTTCCACCTAGGGCTGGTACCTTTAGTGAGGGTTAATTC
GS5	Forward	CATGGAATTAACCCTCACTAAAGGATTTCCCCTTCTATCACGCTCGCGACCTAGGCGTTGGGGCTCGTATGGAAGACGT
	Reverse	CTTCCATACGAGCCCCAACGCCTAGGTCGCGAGCGTGATAGAAGGGGAAATCCTTTAGTGAGGGTTAATTC
GS6	Forward	CATGGAATTAACCCTCACTAAAGGTAACGCCCGGGGTCTGTGTGTCGCTCCCTAAATGGAAGACGT
	Reverse	CTTCCATTTAGGGAGCGACACACAGACCCCGGGCGTTACCTTTAGTGAGGGTTAATTC
GS7	Forward	CATGGAATTAACCCTCACTAAAGGCGGCGCTCCGCGGCCGTTGGAGGGGCCGCCGATGGAAGACGT
	Reverse	CTTCCATCGGCGGCCCCTCCAACGGCCGCGGAGCGCCGCCTTTAGTGAGGGTTAATTC
GS8	Forward	CATGGAATTAACCCTCACTAAAGGCCCGGACGAGCCGGGCCGGCCGGATGGAAGACGT
	Reverse	CTTCCATCCGGCCGGCCCGGCTCGTCCGGGCCTTTAGTGAGGGTTAATTC

3.5.5 Synthesis of Reporter mRNAs In Vitro

RNA synthesis was performed *in vitro* from plasmids pT3-5'UTR-FL-pA containing the test 5'UTR and pT3-RL-pA (Matsuura, Shinmyo, & Kato, 2008) as described previously (Matsuura et al., 2013).

3.5.6 Protoplast Isolation

Three-day-old Oc suspension cells were collected and gently shaken in protoplastization enzyme solution (4% Cellulase RS, 1% macerozyme R10, 0.1% $CaCl_2 \cdot 6\,H_2O$, 0.1% MES, 0.4 M mannitol, pH 5.6) at 30°C for 3 h. The isolation solutions containing crude protoplasts were filtered through a 40-µm nylon sieve, and the same volume of W5 solution (154 mM of NaCl, 125 mM of $CaCl_2$, 5 mM of KCl, 2 mM of Mes-KOH, pH 5.6) was added to the solutions. After centrifugation for 4 min at 800 rpm, pelleted protoplasts were collected and washed once more in the W5 solution by centrifugation. Pelleted protoplasts were added into the W5 solution and incubated on ice for 30 min. The final protoplast density was adjusted to 1×10^6 protoplasts ml^{-1}.

3.5.7 Protoplast Transient Expression Assay

Two µg of capped F-Luc mRNAs harboring a 5'UTR and 0.4 µg of capped R-Luc mRNAs (internal control) were mixed with 1.9×10^6 protoplasts, and an equal volume of polyethylene glycol-CMS (PEG-CMS) solution [200 mM mannitol, 0.1 M $Ca(NO_3)_2$, 40% PEG 4000] was then added to each sample. The protoplast mixture was incubated at room temperature for 20 min, and 1 ml of protoplast medium (400 mM of mannitol supplemented with R2S) was added. The transiently transfected protoplasts were then incubated at 30°C for 20 min, lysed in Passive Lysis Buffer (Promega) and assayed for R-Luc and F-Luc activities by using the Dual-Luciferase Reporter Assay System (Promega, Madison, WI, USA) and a plate reader (TriStar LB 941: Berthold Technologies, Bad Wildbad, Germany).

3.5.8 Evaluation

The results are shown in Figure 3.2. The right figure is a result of a reproductive experiment, i.e., I did the same synthesis experiment twice in order to certain that I did not make mistakes in the first experiment. As shown in Figure 3.2, the correlation coefficients were very high (0.89 and 0.91). Therefore, as predicted, the PR-value became larger, and the true translation efficiency became larger, i.e., the prediction

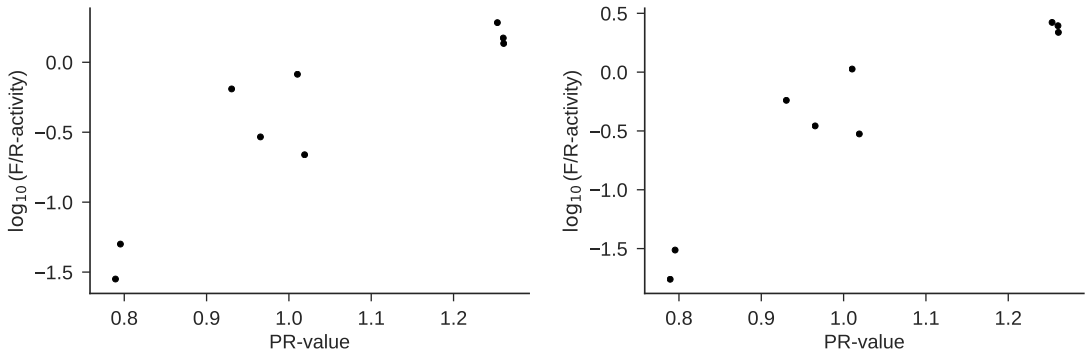

Figure 3.2: Correlation coefficients: x-axis is predicted PR-values, and y-axis is observed $\log_{10}$(F/R-luc activity). Correlation coefficients on left are 0.89, and those on right are 0.91. Right is result of reproductive experiment.

model worked well even for the artificial mRNA. In addition, the two scatter plots were similar to each other; hence, the result of this synthesis experiment is reproducible.

Considering the synthesis experiments, the prediction model built in Section 3.3 can predict the translation efficiencies of artificial mRNAs accurately. Therefore, in Algorithm 1, increasing the number B allows me to obtain translation enhancers.

3.6 Conclusion

I proposed the R-STEINER to generate 5'UTR sequences which increase the amount of translated proteins of certain gene. In the B-step, I built the prediction model of PR-value. The best model to predict PR-value was ensemble model of the three tree-based models: the random forest, the gradient boosting and the XGBoost. This is because the tree-based methods are robust for count features and all of the features, except for secondary energy, engineered in Section 3.3.1 are discrete type features. This result was common between *HS* and *Con*; therefore, the result that the ensemble model is the best prediction model does not depend on the condition. Then, using the rice, I clarified that the prediction model used in the R-STEINER can predict the amount of translated proteins even for artificial mRNA. From this result, it is certain that the prediction model can predict the amount of translated proteins of the 5'UTRs which are generated in G-step. Therefore, the R-STEINER generates the 5'UTRs which actually increase the amount of translated proteins by increasing the iteration B in Algorithm 1. Hence, I do real synthesis experiments for the generated mRNA by R-STEINER and I can reduce the cost, the time and effort.

The point which should be improved of the R-STEINER is the way of sequence generation. In G-step, I generate nucleotides randomly and yield the mRNA sequences by combining the generated nucleotides. By rights, I should solve the optimization problem

$$\boldsymbol{x}' = \arg\max_{\boldsymbol{x}} \hat{h}(\boldsymbol{x}) \tag{3.10}$$

and generate 5'UTR sequence corresponding to $\boldsymbol{x}'$. However, for this approach, I have the following two difficulties. The first difficulty is that solving Equation (3.10) is difficult. The second difficulty is that one feature vector does not correspond to one 5'UTR sequence. As can be seen by Section 3.3.1, one feature vector corresponds to some 5'UTR sequences.

For solving the first problem, I should obtain the method to solve the optimization problem of the function whose variable is 238-dimensional vector and which I cannot write by explicit form. After the first problem is solved, I need a method to generate one sequence by one feature vector. Specifically, I develop a method to select one sequence from candidates of sequences by some kind of criterion. If these two problems are solved, I can generate 5'UTRs which increase the amount of translated proteins of certain gene in shorter time.

In addition to above suggestions, I consider the feature importances which are calculated by the ensemble predicted model. I calculated the feature importance by the sample mean among the three tree-based models: random forest, gradient boosting and XGBoost (see Figure 3.3). These feature importances mean how much each feature contributes to the prediction of PR-value; the feature whose feature importance is high does not always affect positively on the protein production. As can be seen by Figure 3.3, the features of 5'UTR affect on the prediction more strongly than the features of CDS and 3'UTR. This result agree to the previous research (Yamasaki et al., 2018), but the result that the second important feature is the counter of GAC is not mentioned. Perhaps, some features of CDS affect the amount of translated proteins.

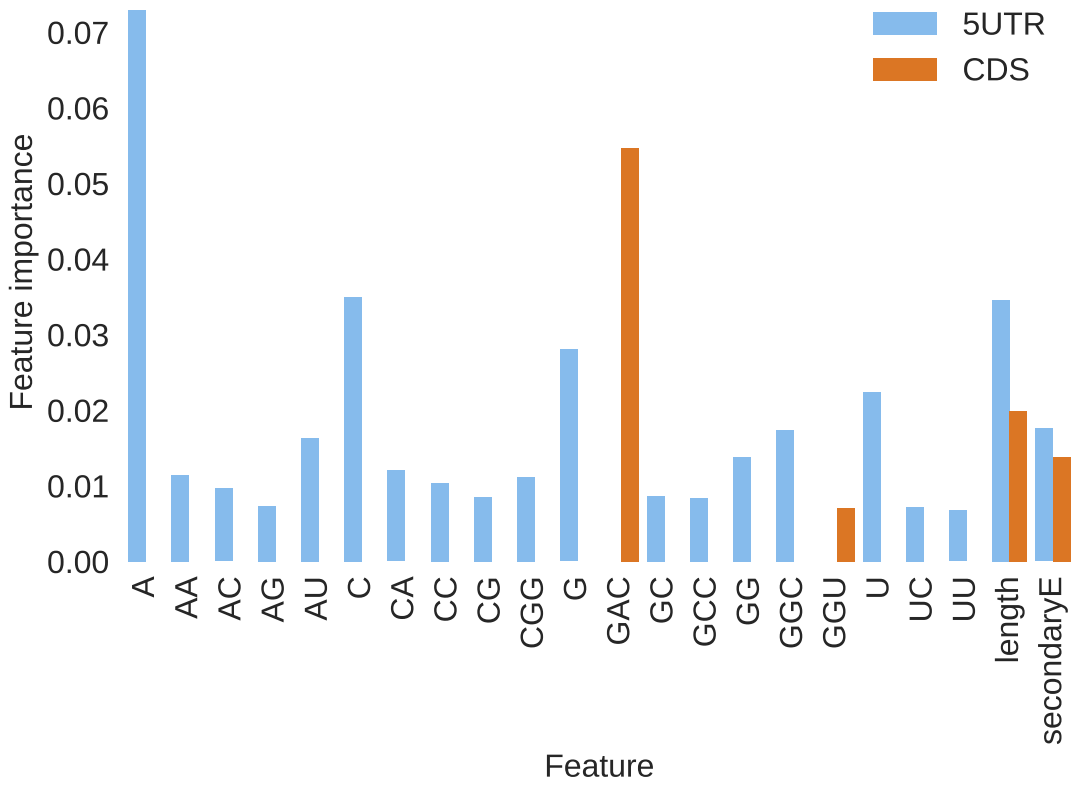

Figure 3.3: Top–90 percentile important features

Chapter 4

TRANS-AM

4.1 Introduction

Machine learning and data mining techniques have been effective in various fields. Specifically, additive tree models, e.g., random forest, gradient boosting and XGBoost, are very effective for predicting an objective variable from an input vector.

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 or changing the class in classification tasks. However, in the real world, we often 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 or decrease the objective variable?"

Let me consider a running example. I am a manager of a service and know that unsatisfied consumers (called low customers) defect to other companies, whereas the consumers with high customer satisfactions use other services in the same company. In such a situation, we should increase the customer satisfactions of low consumers. It is clear that we can predict the customer satisfaction from the input variables[1] of consumers such as quality of service and cost, by using a prediction model. However, generally it is hard for me to find knowledges to increase customer satisfactions of low consumers.

In this chapter, we propose the method, named TRANS-AM. This method enables

[1]In this paper, input variables mean the components of an input vector.

me 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 decision tree, splitting the input space into subspaces, allows me 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, Silvestri, Haines, and Lalmas (2017). Their method is built for classification tasks, and we expanded the method for regression tasks. I also relax a restriction of a decision tree which is supposed in the previous study by Tolomei et al. (2017). Considering the previous running example, the TRANS-AM enables me to find how to deal with consumers by discovering the optimal input vectors satisfying their customer satisfaction.

4.2 Related Work

I first discuss earlier studies on extracting actionable knowledge. Cao, Luo, and Zhang (2007); Hilderman and Hamilton (2000) focused on the development of effective interestingness metrics. Liu and Hsu (1996); Liu, Hsu, and Ma (1999) proposed methods for pruning and summarizing the learnt rules, as well as matching rules by similarity. Cao and Zhang (2007); Cao et al. (2010) 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 my proposed method does not.

I now discuss tree-based methods. Du, Hu, Ling, Fan, and Liu (2011); Karim and Rahman (2013); Yang, Yin, Ling, and Pan (2007); Yang, Yin, Ling, and Chen (2003) 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 (J. Friedman et al., 2001). Our proposed method, on the other hand, can use random forest whose prediction precision is better than that of a decision tree.

Cui, Chen, He, and Chen (2015) proved that the optimal action extraction (OAE) problem similar to the problem we solve in Section 4.3 is generally NP-hard by reducing it to DNF-MAXSAT (Manindra & Thomas, 2000) 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 (CPLEX, 2009). Tolomei et al.

(2017) developed a method of solving the OAE problem with an ε-satisfactory instance, which is explained in Section 4.3. I call this method as actionable feature tweaking (AFT). However, the AFT is for classification tasks and has the following restriction. Once we use the input variable x_1 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. For my study, we expanded their method for regression tasks and relaxed the above restriction.

4.3 TRANS-AM: Proposed Method

In this section, we explain the TRANS-AM for feature transformation to increase or decrease objective variables 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*. I expanded AFT to change the objective variable to *regression* and liberalize one assumption of AFT regarding the root-to-leaf paths of each decision tree.

4.3.1 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 $y \in \mathcal{Y} \subset \mathbb{R}$.

I 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 $T_1, \cdots, T_K$. I 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 $\hat{h}_k$.

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

$$\mathcal{X} = \mathcal{X}_{k,1} \oplus \mathcal{X}_{k,2} \oplus \cdots \oplus \mathcal{X}_{k,M}, \tag{4.1}$$

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

$$\hat{h}_k(\boldsymbol{x}) = \sum_{m=1}^{M} \gamma_{k,m} 1\left[\boldsymbol{x} \in \mathcal{X}_{k,m}\right]. \tag{4.2}$$

4.3.2 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.

I assume that for any fixed decision tree T_k, Assumption 1 holds.

Assumption 1. *For any fixed decision tree T_k of the random forest $\mathcal{T}$, Equation (4.3) is holds.*

$$\exists \mathcal{X}_{k,m} \subset \mathcal{X} \quad s.t. \quad \forall \boldsymbol{x} \in \mathcal{X}_{k,m},\, \hat{h}_k(\boldsymbol{x}) = \gamma_{k,m} \geq t_1 \tag{4.3}$$

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

$$\mathcal{X}_{k,m}^{t_1} = \prod_{i=1}^{d} \left[\theta_i^{\text{low}}, \theta_i^{\text{upp}}\right], \quad \theta_i^{\text{low}}, \theta_i^{\text{upp}} \in \mathbb{R} \cup \{-\infty, \infty\}. \tag{4.4}$$

In Equation (4.4), 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 decision tree T_k from root to m-th leaf is represented as*

$$p_{k,m} = \left\{ \left(x_1 \overset{\geq}{\leq} \theta_1\right), \left(x_2 \overset{\geq}{\leq} \theta_2\right), \cdots, \left(x_d \overset{\geq}{\leq} \theta_d\right) \right\}. \tag{4.5}$$

Assumption 2 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.

4.3.3 ε-Satisfactory Instance

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

$$\mathscr{X}_k = \{ \mathcal{X}_{k,m} \mid \forall \boldsymbol{x} \in \mathcal{X}_{k,m},\, \hat{h}_k(\boldsymbol{x}) = \gamma_{k,m} \geq t_1 \}, \tag{4.6}$$

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, $|\mathscr{X}|$ 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.

$$\boldsymbol{x}^{t_1}_{k(\varepsilon)}[i] = \begin{cases} \theta_i^{\text{upp}} - \varepsilon & \theta_i^{\text{low}} = -\infty \\ \theta_i^{\text{low}} + \varepsilon & \theta_i^{\text{upp}} = \infty \\ \frac{\theta_i^{\text{low}} + \theta_i^{\text{upp}}}{2} & \text{otherwise} \end{cases} \tag{4.7}$$

In Equation (4.7), ε means the distance between $\boldsymbol{x}^{t_1}_{k(\varepsilon)}$ and the boudaries of subspace, i.e, the position of $\boldsymbol{x}^{t_1}_{k(\varepsilon)}$ is determined by ε if $\theta_i^{\text{low}} = -\infty$ or $\theta_i^{\text{upp}} = \infty$.

The ε-satisfactory instance defined by Equation (4.7) belongs to $\mathcal{X}^{t_1}_{k,m} \in \mathscr{X}$. If $\boldsymbol{x}^{t_1}_{k(\varepsilon)}[i] = \theta_i - \varepsilon$, the i-th element of $\boldsymbol{x}^{t_1}_{k(\varepsilon)}$ is the point transformed by $-\varepsilon$ from θ_i for the negative direction parallel to the x_i axis (or vice versa). If $\boldsymbol{x}^{t_1}_{k(\varepsilon)}[i] = \left(\theta_i^{\text{low}} + \theta_i^{\text{upp}}\right)/2$, the i-th element of $\boldsymbol{x}^{t_1}_{k(\varepsilon)}$ is the centre of interval $\left[\theta_i^{\text{low}}, \theta_i^{\text{upp}}\right]$. In any case $\boldsymbol{x}^{t_1}_{k(\varepsilon)} \in \mathcal{X}^{t_1}_{k,m}$ holds; therefore, $\hat{h}\left(\boldsymbol{x}^{t_1}_{k(\varepsilon)}\right) \geq t_1$ also holds.

4.3.4 Feature Transformation

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

$$\mathcal{X}^{t_1}_{\cdot,\cdot} = \bigcup_{k=1}^{K} \bigcup_{m:\mathcal{X}^{t_1}_{k,m} \in \mathscr{X}} \boldsymbol{x}^{t_1}_{k(\varepsilon)}, \tag{4.8}$$

where $\boldsymbol{x}^{t_1}_{k(\varepsilon)}$ depends on subspace $\mathcal{X}^{t_1}_{k,m}$. Therefore, $\bigcup_{m:\mathcal{X}^{t_1}_{k,m} \in \mathscr{X}} \boldsymbol{x}^{t_1}_{k(\varepsilon)}$ 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.

$$\boldsymbol{x}' = \underset{\boldsymbol{x}^{t_1}_{k(\varepsilon)} \in \mathcal{X}^{t_1}_{\cdot,\cdot} | \hat{f}\left(\boldsymbol{x}^{t_1}_{k(\varepsilon)}\right) \geq t_1}{\arg\min} \delta\left(\boldsymbol{x}, \boldsymbol{x}^{t_1}_{k(\varepsilon)}\right) \tag{4.9}$$

In Equation (4.9), cost function δ means the distance between $\boldsymbol{x}$ and $\boldsymbol{x}^{t_1}_{k(\varepsilon)}$. 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 Section 4.1, Cui et al. (2015) proved that the OAE problem, which is essentially identical to Equation (4.9), 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. (2017) 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. (2017) to discovery $\boldsymbol{x}'$. I show the algorithm for finding $\boldsymbol{x}'$ in Algorithm 2. This algorithm seek $\boldsymbol{x}'$ solution

of the following optimization problem:

$$\boldsymbol{x}' = \underset{\boldsymbol{x}^{t_1}_{k(\varepsilon)} \in \mathcal{X}^{t_1}_{\cdot,\cdot}}{\arg\min}\ \delta\left(\boldsymbol{x}, \boldsymbol{x}^{t_1}_{k(\varepsilon)}\right). \tag{4.10}$$

In Equation (4.10), we drop the condition $\hat{f}\left(\boldsymbol{x}^{t_1}_{k(\varepsilon)}\right) \geq t_1$ of Equation (4.9), because the condition is implicitly satisfied by definition of ε-satisfactory instance. Algorithm 2 often returns $\boldsymbol{x}' = \boldsymbol{x}$ because sometimes all the ε-satisfactory instances $\boldsymbol{x}^{t_1}_{j(\varepsilon)}$ built with Algorithm 2 do not satisfy the 8-th line if-statement $\hat{f}\left(\boldsymbol{x}^{t_1}_{j(\varepsilon)}\right) \geq t_1$. Hence we should evaluate the TRANS-AM carefully in Section 4.4.

Algorithm 2 Algorithm of TRANS-AM

Require: $\mathcal{T} = \{T_1, T_2, \cdots, T_K\}$, thresholds t_0, t_1, feature vector $\boldsymbol{x}$ such that $f(\boldsymbol{x}) < t_0$, cost function δ, 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** $\hat{f}(\boldsymbol{x}) < t_1 \wedge \hat{h}_k(\boldsymbol{x}) < t_1$ **then**
5: make $\mathscr{X}_k$
6: **for** $\mathcal{X}^{t_1}_{k,m} \in \mathscr{X}_k$ **do**
7: build ε-satisfactory instance $\boldsymbol{x}^{t_1}_{j(\varepsilon)}$
8: **if** $\hat{f}\left(\boldsymbol{x}^{t_1}_{j(\varepsilon)}\right) \geq t_1$ **then**
9: **if** $\delta\left(\boldsymbol{x}, \boldsymbol{x}^{t_1}_{j(\varepsilon)}\right) < \delta_{\min}$ **then**
10: $\boldsymbol{x}' \leftarrow \boldsymbol{x}^{t_1}_{j(\varepsilon)}$
11: $\delta_{\min} \leftarrow \delta\left(\boldsymbol{x}, \boldsymbol{x}^{t_1}_{j(\varepsilon)}\right)$
12: **end if**
13: **end if**
14: **end for**
15: **end if**
16: **end for**
17: **return** $\boldsymbol{x}'$

4.4 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 2 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 2 does not always satisfy $f(\boldsymbol{x}') \geq t_1$, i.e., Algorithm 2 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 2 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 simulation.

As we mentioned in Section 4.3, Algorithm 2 often returns $\boldsymbol{x}' = \boldsymbol{x}$ Therefore, we use Equation (4.17) as an indicator for evaluating this problem.

4.4.1 Experimental Setting

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

Step 1. generate $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N \overset{\text{i.i.d.}}{\sim} \mathcal{N}_d(\mathbf{1}_d, I_d)$.

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 Section 4.3. In my simulation, we used the following functions as $f(\boldsymbol{x})$.

$$f_1(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{x} \tag{4.11}$$

$$f_2(\boldsymbol{x}) = \sin\left(\boldsymbol{a}^T \boldsymbol{x}\right) \tag{4.12}$$

$$f_3(\boldsymbol{x}) = \sum_{i=1}^{n} a_i \sin x_i \tag{4.13}$$

$$f_4(\boldsymbol{x}) = \exp\left(\boldsymbol{a}^T \boldsymbol{x}\right) \tag{4.14}$$

$$f_5(\boldsymbol{x}) = \frac{1}{1 + \exp\left(\boldsymbol{a}^T \boldsymbol{x}\right)} \tag{4.15}$$

In functions Equation (4.11) - Equation (4.15), $\boldsymbol{a} \sim \mathcal{N}(\mathbf{0}, I)$ 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 Equation (4.16), Equation (4.17), and Equation (4.18).

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 2 often yields the same vector as an input vector. Therefore, we evaluated how many yielded vectors satisfy my purpose $f(\boldsymbol{x}') \geq t_1$ per number of transformed vectors with the R score.

$$P = \frac{|\{\, \boldsymbol{x}' \mid f(\boldsymbol{x}') \geq t_1 \,\}|}{\left|\{\, \boldsymbol{x} \mid \hat{f}(\boldsymbol{x}) < t_0 \,\}\right|} \tag{4.16}$$

$$Q = \frac{|\{\, \boldsymbol{x}' \mid \boldsymbol{x}' \neq \boldsymbol{x} \,\}|}{\left|\{\, \boldsymbol{x} \mid \hat{f}(\boldsymbol{x}) < t_0 \,\}\right|} \tag{4.17}$$

$$R = \frac{|\{\, \boldsymbol{x}' \mid f(\boldsymbol{x}') \geq t_1 \,\}|}{|\{\, \boldsymbol{x}' \mid \boldsymbol{x}' \neq \boldsymbol{x} \,\}|} \tag{4.18}$$

Among these three scores, a relationship $P = QR$ holds. Score P is most noticeable score and scores Q and R construct P. I simulated all combinations of $\varepsilon \in \{\, 0.01, 0.05, 0.1, 0.5, 1, 1.5 \,\}$, $N_0 = 1000$, $N_1 = 250$ and $d = 50$. A sample size of $N_0 = 1000$ is reasonable. For example, (Zhang, Li, Yu, & Tian, 2016; Breiman, 2001) used a dataset with a sample size of 1000.

4.4.2 Result and Consideration

I show the simulation results in Figure 4.1, Figure 4.2, and Figure 4.3. Figure 4.1 shows the relationship between ε and P score, Figure 4.2 shows the relationship between ε and Q score, and Figure 4.3 shows the relationship between ε and Q score. In the each figure, *linear, sin, sinsum, exp*, and *logistic* mean the datasets generated by Equation (4.11), Equation (4.12), Equation (4.13), Equation (4.14), and Equation (4.15), respectively.

As shown in Figure 4.1, for $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, the dataset generated by Equation (4.15) shows the best P-score. In addition, Equation (4.12) shows the second-best P-score. This is because Equation (4.15) and Equation (4.12) are upper bounded, i.e., $\exists K$: const. s.t. $\forall \boldsymbol{x} \in \mathbb{R}$, $f(\boldsymbol{x}) < K$. As we explained in Section 4.3, 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.

On the other hand, in the centre and right figures of Figure 4.1, *logistic* becomes worse. Such a change is caused by the shape of function Equation (4.15). The exponential function $g(x) = \exp(x)$ satisfies $\lim_{x \to \infty} = 1$ and $\lim_{x \to -\infty} = 0$, and obviously for almost all $x \in \mathbb{R}$, $g(x) \simeq 1$ or $g(x) \simeq 0$. By such a property of the exponential function, percentile points above 70% are pulled in the positive direction because the objective variables generated by Equation (4.15) depend on the noise ε more strongly than those by the others. In fact, the distribution of $f(\boldsymbol{x}')$ is illustrated in Figure 4.4. As shown in Figure 4.4, the threshold is pulled in the positive direction as percentile point becomes larger. However, the density function of the objective variables of *logistic* tend to be high near the area satisfying $f(\boldsymbol{x}') = 1$ as ε increases. Therefore, if the threshold is not high increasing ε must lead to good performance.

As shown in Figure 4.2, in the left and centre figures, each R score is higher than 0.8, except for *exp*. These results mean that for the datasets generated by Equation (4.14), Algorithm 2 cannot transform $\boldsymbol{x}$ into $\boldsymbol{x}'$. This is also because random forest cannot approximate Exponential Function Equation (4.14) well. Of course, other functions Equation (4.11) and Equation (4.13) are not upper bounded. However, Exponential Function Equation (4.14) diverges to infinity earlier than Equation (4.11) and Equa-

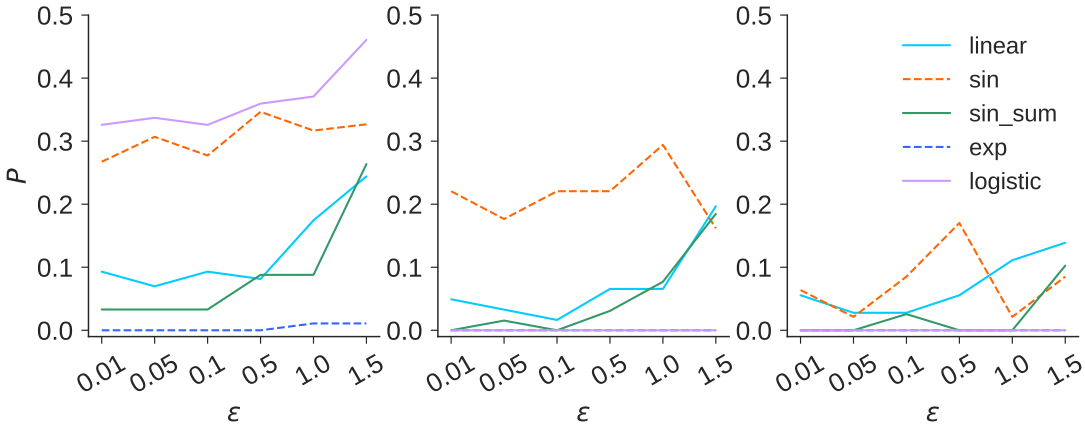

Figure 4.1: Relationships between ε and P. Left: $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, Centre: $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$, Right: $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

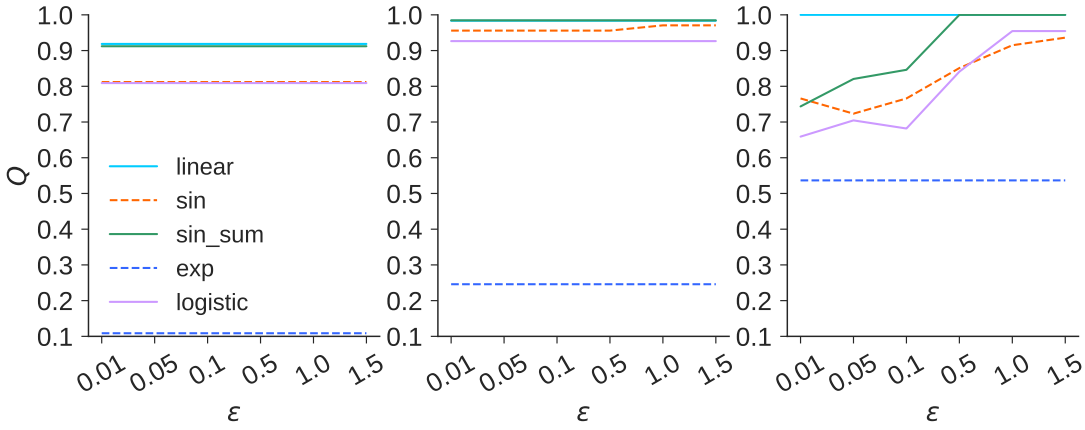

Figure 4.2: Relationships between ε and Q. Left: $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, Centre: $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$, Right: $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

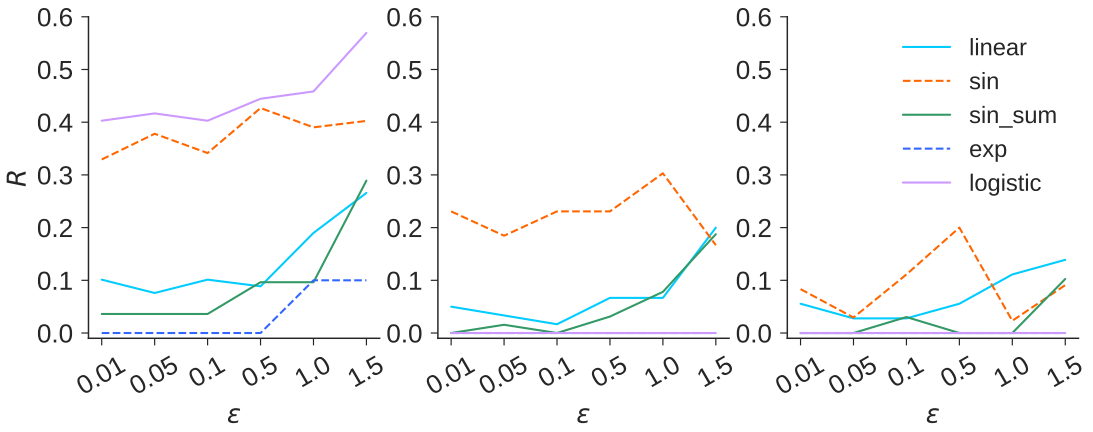

Figure 4.3: Relationships between ε and R. Left: $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, Centre: $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$, Right: $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

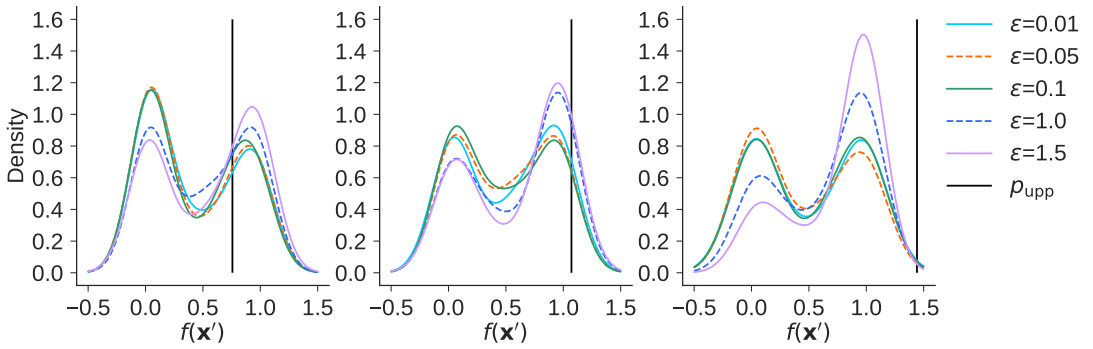

Figure 4.4: Distributions of *logistic*. Left: $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, Centre: $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$, Right: $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

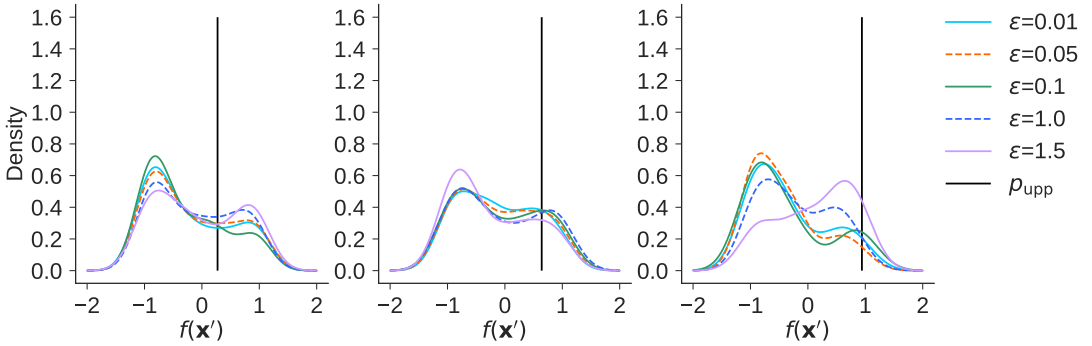

Figure 4.5: Distributions of objective variables corresponding to transformed input variables. Left: $(p_{\text{low}}, p_{\text{upp}}) = (40, 60)$, Centre: $(p_{\text{low}}, p_{\text{upp}}) = (30, 70)$, Right: $(p_{\text{low}}, p_{\text{upp}}) = (20, 80)$

tion (4.13). 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 aproximation precision for the dataset generated by Equation (4.14) 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 2 cannot be satisfied, i.e., $\boldsymbol{x}$ is yielded using Algorithm 2. 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 Figure 4.2, turning the ε well, the Q-scores become higher than 0.8.

As shown in the left figure of Figure 4.3, the R scores of *logistic* and *sin* can be greater than 0.4 by turning ε for $p_{\text{low}} = 40$, $p_{\text{upp}} = 60$. In the centre and right figures of Figure 4.3, however, the R score of *logistic* is one of the lowest. The reason is referred to in the discussion of Figure 4.1. As p_{low} becomes smaller and p_{upp} becomes larger, the R score decrease.

Figure 4.5 illustrates the reason the TRANS-AM performs well for the dataset generated by Equation (4.12). The thresholds p_{upp} for *sin* are not pulled in the positive direction so strongly, as for *logistic*. Therefore, the TRANS-AM performed better for the dataset generated by Equation (4.12) than for the other datasets.

4.5 Conclusion

I proposed the TRANS-AM for discovering new input vectors to increase or decrease the objective variables in a regression task. I relaxed the restrictions assumed by Tolomei et al. (2017) 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 vectors to improve the objective variables?" The TRANS-AM is an answer for this question and we have disclosed the situation that the method works well.

I evaluated the TRANS-AM and we found that for the dataset whose objective variables are generated by the sin formula, the TRANS-AM shows good precision on the whole. This is because the thresholds in each case of Figure 4.5 are not pulled in some kind of direction. On the other hand, for the dataset whose objective variables are generated by the logistic formula, the TRANS-AM does not work well except when $p_{\text{low}} = 40$ and $p_{\text{upp}} = 60$. In such a situation, it is hard for TRANS-AM to find the transformed input vector satisfying the condition. This is because the upper 70 percentile points are pulled in the positive direction. Even for the dataset whose objective variables are generated by Equation (4.15) the TRANS-AM works well, if we choose a suitable threshold.

For future works, we plan to extend the TRANS-AM to gradient boosting and XGBoost. 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.

Chapter 5

Conclusion

In this paper, I proposed the R-STEINER for generating 5'UTRs which allow a certain gene to increase the amount of translated proteins as well as the TRANS-AM for discovering new input vectors to increase or decrease the objective variables in a regression task. The conclusions of each method are described in Chapter 3 and Chapter 4. In this chapter, I discuss applications of the TRANS-AM to the 5'UTR generation and the future of this study.

5.1 Application of TRANS-AM to 5'UTR Generation

As I described in Chapter 1, some companies have patents of their sequences and require a technique to improve the ability of their sequences in order to increase the amount of translated proteins, i.e., they want to edit their sequences in order to increase the amount of translated proteins by minimum editing cost. As a solution to this problem, I will propose a method which will be established by using TRANS-AM. Specifically, I will obtain a feature vector whose corresponding PR-value is greater than some threshold by TRANS-AM. Then I will generate a 5'UTR sequence according to this feature vector.

For this method, I have to solve some problems. First, if I apply the TRANS-AM to generate the feature vector, I cannot deal with the prediction model built in R-STEINER. This is because the prediction model is an ensemble model of three models. The TRANS-AM can deal with only random forest. Therefore I have to build the prediction model only by random forest or expand the TRANS-AM so that I can use it in the prediction model built by other tree-based prediction models.

Second, I can use only the features calculated from 5'UTR sequences because ε in TRANS-AM impinge on all of the components of the feature vector. As I cannot

control the sequences of CDS and 3'UTR, I can use the features of the 5'UTR sequence. For solving this problem, I have to expand the TRANS-AM so that I can select the transformed features or build the prediction model showing similar prediction accuracy to my prediction model.

Finally, as mentioned in Section 3.6, I do not have the way to obtain 5'UTR sequence minimizing the Levenshtein distance. It is true that the Bray-Curtis distance correlates strongly with the Levenshtein distance, but I cannot reconstruct the 5'UTR sequence by the feature vector. No effective solutions have been proposed yet for this problem except for manpower, i.e. I select the point which should be edited and change the sequence according to the difference between the original feature vector and the new one.

5.2 Future Work

As described in Chapter 1, my study is useful for the drug development by plants. R-STEINER enables me to answer the question "What kind of 5'UTR sequence should I synthesize?" without real experiments. Real experiments to synthesize 5'UTRs need much cost, time and effort. If I use actual trees for the experiments, I may need a period of one year or more. Therefore, my method which allows me to answer the above question is very useful for the experiments; i.e., I synthesize only 5'UTRs obtained by R-STEINER, resulting in the reduction of the cost, time and effort.

Solving problems mentioned in Section 5.1 enables me to obtain the translation enhancer by utilizing existing 5'UTR sequences. It is supposed that the synthesized 5'UTR based on the existing 5'UTR is *stable*; in real experiments, various factors can affect the translation of mRNA, therefore artificial 5'UTRs sometimes cause some trouble, e.g., the rupture of the sequence. In addition, this meets the requirement that some companies want to utilize their 5'UTR sequences protected by their patents. By synthesizing the 5'UTR sequences which are likely to be stable, I can reduce the time, cost and effort—synthesizing astable 5'UTR is a waste of time, cost and effort—.

At all events, my proposed methods allow me to reduce the cost, time and effort of real synthesis experiment. More efficient synthesis experiments lead to the efficient drug development by plants.

References

Alberts, B., Bray, D., Hopkin, K., Johnson, A., Lewis, J., Raff, M., ... Walter, P. (2013). *Essential cell biology*. Garland Science.

Baba, A., Hasezawa, S., & Syōno, K. (1986). Cultivation of rice protoplasts and their transformation mediated by agrobacterium spheroplasts. *Plant and cell physiology*, *27*(3), 463–471.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Cao, L., Luo, D., & Zhang, C. (2007). Knowledge actionability: satisfying technical and business interestingness. *IJBIDM*, *2*, 496-514.

Cao, L., & Zhang, C. (2007). Domain-driven, actionable knowledge discovery. *IEEE Intelligent Systems*, *22*(4).

Cao, L., Zhao, Y., Zhang, H., Luo, D., Zhang, C., & Park, E. K. (2010). Flexible frameworks for actionable knowledge discovery. *IEEE Transactions on Knowledge and Data Engineering*, *22*(9), 1299–1312.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

CPLEX, I. I. (2009). V12. 1: User's manual for cplex. *International Business Machines Corporation*, *46*(53), 157.

Cui, Z., Chen, W., He, Y., & Chen, Y. (2015). 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).

Du, J., Hu, Y., Ling, C. X., Fan, M., & Liu, M. (2011). Efficient action extraction with many-to-many relationship between actions and features. In *International workshop on logic, rationality and interaction* (pp. 384–385).

Friedman, J., Hastie, T., Höfling, H., Tibshirani, R., et al. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, *1*(2), 302–332.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning.*

Springer series in statistics New York.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Fujiyama, K. (n.d.). Challenge for medical protein production by using plant.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Hilderman, R. J., & Hamilton, H. J. (2000). Applying objective interestingness measures in data mining systems. In *European conference on principles of data mining and knowledge discovery* (pp. 432–439).

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Karim, M., & Rahman, R. M. (2013). Decision tree and naive bayes algorithm for classification and generation of actionable knowledge for direct marketing. *Journal of Software Engineering and Applications*, *6*(04), 196.

Kawaguchi, R., & Bailey-Serres, J. (2005). mrna sequence features that contribute to translational regulation in arabidopsis. *Nucleic Acids Research*, *33*(3), 955–965.

Liu, B., & Hsu, W. (1996). Post-analysis of learned rules. In *Aaai/iaai, vol. 1* (pp. 828–834).

Liu, B., Hsu, W., & Ma, Y. (1999). Pruning and summarizing the discovered associations. In *Proceedings of the fifth acm sigkdd international conference on knowledge discovery and data mining* (pp. 125–134).

Lorenz, R., Bernhart, S. H., Zu Siederdissen, C. H., Tafer, H., Flamm, C., Stadler, P. F., & Hofacker, I. L. (2011). Viennarna package 2.0. *Algorithms for Molecular Biology*, *6*(1), 26.

Manindra, A., & Thomas, T. (2000). *Satisfiability problems* (Tech. Rep.). Technical Report.

Matsuura, H., Shinmyo, A., & Kato, K. (2008). Preferential translation mediated by hsp81-3 5'-utr during heat shock involves ribosome entry at the $5'$ -end rather than an internal site in arabidopsis suspension cells. *Journal of bioscience and bioengineering*, *105*(1), 39–47.

Matsuura, H., Takenami, S., Kubo, Y., Ueda, K., Ueda, A., Yamaguchi, M., ... Kato, K. (2013). A computational and experimental approach reveals that the $5'$ -proximal region of the $5'$ -utr has a cis-regulatory signature responsible for heat stress-regulated mrna translation in arabidopsis. *Plant and cell physiology*, *54*(4),

474–483.

Maxmen, A., et al. (2012). Drug-making plant blooms. *Nature*, *485*(7397), 160.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Močkus, J. (1975). On bayesian methods for seeking the extremum. In *Optimization techniques ifip technical conference* (pp. 400–404).

Mockus, J. (1977). On bayesian methods for seeking the extremum and their application. In *Ifip congress* (pp. 195–200).

Mockus, J. (2012). *Bayesian approach to global optimization: theory and applications* (Vol. 37). Springer Science & Business Media.

Rao, C. R., Rao, C. R., Statistiker, M., Rao, C. R., & Rao, C. R. (1973). *Linear statistical inference and its applications* (Vol. 2). Wiley New York.

Rastrigin, L. (1963). The convergence of the random search method in the extremal control of a many parameter system. *Automaton & Remote Control*, *24*, 1337–1342.

Saotome, A., Kimura, S., Mori, Y., Uchiyama, Y., Morohashi, K., & Sakaguchi, K. (2006). Characterization of four recq homologues from rice (oryza sativa l. cv. nipponbare). *Biochemical and biophysical research communications*, *345*(4), 1283–1291.

Schrack, G., & Choit, M. (1976). Optimized relative step size random searches. *Mathematical Programming*, *10*(1), 230–244.

Schumer, M., & Steiglitz, K. (1968). Adaptive step size random search. *IEEE Transactions on Automatic Control*, *13*(3), 270–276.

Seber, G. A., & Lee, A. J. (2012). *Linear regression analysis* (Vol. 936). John Wiley & Sons.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., ... Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, *56*(1), 116–124.

Sugio, T., Satoh, J., Matsuura, H., Shinmyo, A., & Kato, K. (2008). The 5'-untranslated region of the oryza sativa alcohol dehydrogenase gene functions as a translational enhancer in monocotyledonous plant cells. *Journal of bioscience and bioengineering*, *105*(3), 300–302.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Tjandra, A., Sakti, S., & Nakamura, S. (2017). Compact recurrent neural network based on tensor train for polyphonic music modeling.

Tolomei, G., Silvestri, F., Haines, A., & Lalmas, M. (2017). 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).

Tyree, S., Weinberger, K. Q., Agrawal, K., & Paykin, J. (2011). Parallel boosted regression trees for web search ranking. In *Proceedings of the 20th international conference on world wide web* (pp. 387–396).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, *57*(2), 137–154.

Wold, H. (1966). *Estimation of principal components and related models by iterative least squares.* New York: Academic Press.

Wold, H. (1985). Partial least squares. *Encyclopedia of statistical sciences*.

Wold, S., Sjöström, M., & Eriksson, L. (2001). Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, *58*(2), 109–130.

Yamasaki, S. (2016). *Research on mechanism of mrna translation in arabidopsis thaliana* (Unpublished doctoral dissertation). Nara Institute of Science and Technology.

Yamasaki, S., Matsuura, H., Demura, T., & Kato, K. (2015). Changes in polysome association of mrna throughout growth and development in arabidopsis thaliana. *Plant and Cell Physiology*, *56*(11), 2169–2180.

Yamasaki, S., Sanada, Y., Imase, R., Matsuura, H., Ueno, D., Demura, T., & Kato, K. (2018). Arabidopsis thaliana cold-regulated 47 gene 5'-untranslated region enables stable high-level expression of transgenes. *Journal of Bioscience and Bioengineering*.

Yamasaki, S., Ueda, K., & Kato, K. (2013). Development of plant expression vector on considering of environmental stress. *Seibutsu-Kogaku kaishi*, *91*(8), 456–460.

Yang, Q., Yin, J., Ling, C., & Pan, R. (2007). Extracting actionable knowledge from decision trees. *IEEE Transactions on Knowledge and data Engineering*, *19*(1), 43–56.

Yang, Q., Yin, J., Ling, C. X., & Chen, T. (2003). Postprocessing decision trees

to extract actionable knowledge. In *Data mining, 2003. icdm 2003. third ieee international conference on* (pp. 685–688).

Yao, J., Weng, Y., Dickey, A., & Wang, K. Y. (2015). Plants as factories for human pharmaceuticals: applications and challenges. *International journal of molecular sciences*, *16*(12), 28549–28565.

Ye, J., Wang, L., Li, G., Chen, D., Zhe, S., Chu, X., & Xu, Z. (2017). Learning compact recurrent neural networks with block-term tensor decomposition. *arXiv preprint arXiv:1712.05134*.

Zhang, C., Li, Y., Yu, Z., & Tian, F. (2016). A weighted random forest approach to improve predictive performance for power system transient stability assessment. In *Power and energy engineering conference (appeec), 2016 ieee pes asia-pacific* (pp. 1259–1263).

Publication List

Journal

Under Review

[1] Tanaka, H., Suzuki Y., Yamasaki, S., Yoshino, K., Kato, K., Nakamura, S. (2018). R-STEINER: Generation Method of 5'UTR for Increasing the Amount of Translated Proteins. *ISJ Transactions on Bioinformatics*

[2] Naito K., Inge, K., Tanaka, H. (2018). Kernel Naive Bayes Discrimination for High-Dimensional Pattern Recognition. *Annals of the Institute of Statistical Mathematics*

International Conference

Under review

[1] Tanaka, H., Suzuki Y., Yoshino, K., Nakamura S. (2018). TRANS-AM: Descovery Method of Optimal Input Vectors Corresponding to Objective Variables. *20th International Conference on Big Data Analytics and Knowledge Discovery* (DaWaK 2018)

Others

Technical Report

[1] Tanaka, H., Suzuki, Y., Yamasaki, S., Kato, K., Yoshino, K., Nakamura, S. (2017). Prediction of Translation Efficiency from mRNA by Using Ensemble Learning. *IPSJ Technical Report on Bioinformatics*

[2] Tanaka, H., Suzuki, Y., Yamasaki, S., Yoshino, K., Kato, K., Nakamura, S. (2018).

R-STEINER: Generation Method of 5'UTR for Making mRNA's Translation Active. *Data Engineering and Information Management Forum 2018*

[3] Tanaka, H., Suzuki, Y., Yamasaki, S., Yoshino K., Kato, K., Nakamura, S. (2018). R-STEINER: Generation Method of 5'UTR for Increasing the Amount of Translated Proteins. *IPSJ Technical Report on Bioinformatics*

Presentation

[1] Tanaka, H., Suzuki, Y., Yamasaki, S., Kato, K., Yoshino, K., Nakamura, S. (2017). Prediction of Translation Efficiency from mRNA by Using Ensemble Learning. *IPSJ Technical Report on Bioinformatics*

[2] Tanaka, H. (2017). Situation Classification for Prediction of Pitching Pattern. *WebDB Forum 2018*

[3] Tanaka, H., Suzuki, Y., Yamasaki, S., Kato, K., Yoshino, K., Nakamura, S. (2017). Generating mRNA with High Translation Efficiency by Using Machine Learning. *Kansai Database Workshop 2017*

[4] Tanaka, H., Suzuki, Y., Yamasaki, S., Yoshino, K., Kato, K., Nakamura, S. (2018). R-STEINER: Generation Method of 5'UTR for Making mRNA's Translation Active. *Data Engineering and Information Management Forum 2018*

[5] Tanaka, H., Suzuki, Y., Yamasaki, S., Yoshino K., Kato, K., Nakamura, S. (2018). R-STEINER: Generation Method of 5'UTR for Increasing the Amount of Translated Proteins. *IPSJ Technical Report on Bioinformatics*