Development of Segmentation Algorithms for Identifying Smartphone Applications with Sustainable Popularity Ranking

March 2020

SEKPON SESSIME BAKUMYNA

Development of Segmentation Algorithms for Identifying Smartphone Applications with Sustainable Popularity Ranking

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2020

SEKPON SESSIME BAKUMYNA

Acknowledgment

I gratefully acknowledge the guidance and support of my academic advisor, Professor Ushio Sumita, who gave me his fullest support at every step of my student life and showed me the way to the dynamic world of Business Analytics. Professor Sumita dedicated numerous hours guiding me throughout my research, even after his retirement, and led me to the completion of this thesis. Without his strict instructions, this thesis could not have been completed. I have learned from him many important values useful not only for my intellectual knowledge but also for my personal life. I would like to express my heartfelt gratitude to Professor Yoshiaki Osawa for the continuous support and guidance he gave me since the day I started my life at the University of Tsukuba as a research student. During the past couple of years, Professor Osawa spent hours discussing my research and kept encouraging me to do my best whenever I was tempted to give up.

I am thankful to my thesis advisors Professor Mamoru AMEMIYA, Professor Mika SATO-ILIC and Professor Akiko YOSHISE, for providing me valuable advice for improving my thesis. Their constructive suggestions have profoundly contributed to improving this thesis up to this version.

I am thankful to the staff members of Shako Office for always being ready to give me their full attention, for their wonderful way of problem-solving and to the University of Tsukuba for assisting me and facilitating my student life.

I thankfully remember my laboratory members and also, the previous members of Sumita Laboratory and Osawa Laboratory for their kind support. I thank my friends in Tsukuba and overseas for their support and encouragement.

My heartfelt gratitude goes to my family worldwide for their loving support, prayers and care throughout my life and especially the completion of this thesis. Without that generous support from every one of you, this thesis could never have been completed.

Abstract

The markets of digital products and services offered through the Internet have been rapidly growing. However, the study of such markets is rather limited in the literature largely because such products and services may be installed and uninstalled dynamically over time in a repeated manner, and consequently it is quite difficult to acquire real data. In addition, financial performance measures of such digital products services are hard to come by. Recently, a Japanese software development company named Fuller, Inc. developed an Android smartphone application named 'Mr. Mobile, the battery saver,' which enabled the company to obtain actual data concerning installments and un-installments of smartphone applications via the legitimate agreement between the company and the users of this application.

Based on the large volume of data acquired from Fuller, Inc. and the ranking information provided by Google in place of financial performance measures, the purpose of this thesis is to fill the above gap by developing segmentation algorithms for identifying "smartphone applications with sustainable popularity ranking". Formally, let $t_0(a)$ be the month in which application 'a' is introduced into the market. Then application 'a' is said to be with sustainable popularity ranking associated with (M, m_0, m_1) if it is ranked within top M by Google Play at least once during the first m_0 months since $t_0(a) + 1$, and is still ranked within top M at least once during the m_0 months after $t_0(a) + 1 + m_1$.

So as to identify such smartphone applications, we first develop segmentation

algorithms using Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Neural Network (NNet) and Naïve Bayes Classifier (NBC) provided by the R language. More specifically, two new segmentation algorithms are constructed by combining the results of the individual six methods.

While the two segmentation algorithms show excellent performance for identifying those smartphone applications with sustainable popularity ranking, they may not specify the key factors behind the segmentation. In order to overcome this pitfall, we secondly focus on LR and DT and develop five different segmentation algorithms, thereby enabling one to detect the driving force behind the segmentation.

These segmentation algorithms are applied to mobile phone applications in the category of Japanese free games, and reveal some subtle characteristics of those free games with sustainable popularity ranking, providing useful information for software development companies.

Table of Contents

List of Figures

List of Tables

Notation

A	Whole set of mobile applications under consideration
a	Mobile application under consideration
D	Whole set of devices under consideration
$D(T)$	Set of devices registered in T
$FA(t)$ $\in T \setminus \{0\}$	Set of applications that appeared for the first time in month t
$I(a,d,t)$	$= \begin{cases} 1 & \text{if } a \in \mathrm{A}(T) \text{ is present in } d \in D(T) \text{ in month } t \in T \\ 0 & \text{else} \end{cases}$
$T = \{0,..,\mathrm{T}\}$	Period of time covered by the dataset (7/2013 to 4/ 2016)
$t_0(a)$	The month in which $a \in A(T)$ appeared for the first time in T
$Rank\ (a,t)$	Ranking of application $a \in A(T)$ in month $t \in T$
FGC-ALL	Japanese Free Games Category
FGC-RP	Japanese Free Games Role Play Subcategory
FGC-CAS	Japanese Free Game Casual Subcategory
FGC-PZ	Japanese Free Game Puzzle Subcategory
FGC-ACT	Japanese Free Game Action Subcategory

Chapter 1

Introduction

1.1 Background of the Study

Supported by the wide spread of mobile devices to access the Internet in a ubiquitous manner from anywhere at any time, the smartphone application market has been growing quite rapidly. According to eMarketer (2016), the population of smartphones all over the world was expected to be over 2.16 billion in 2016, and to reach 2.56 billion in 2018. The number of smartphone users worldwide in 2019 surpasses 3 billion and is forecast to further grow by several hundred million in the next few years (Statista, 2019).

For the Japanese market, it is reported in eMarketer (2016) that the smartphone audience of 82.2 million in 2015 would exceed 88.9 million by the end of 2017, with a rise from 64.8% to 70.3% of the whole Japanese population. This rise reached 70,8% in 2018 and is said to over 76% in 2022, according to Smaato (2018).

The smartphone population has grown in such a rapid speed because of a variety of enriched applications, including those in such areas as Finance, Businesses, Communications, Music, Movies, Sports, Dictionaries, Education and Games. In a report by AppBrain (2019), the number of Android applications available on Google Play (2019), which is the official application store for Android OS, was more than 2.8 million in December 2019. It indicates that Games constitute the largest portion of smartphone applications, having 13% of the total number of applications. The report

further shows that Free Games amount to almost 94% of Games.

To the best knowledge of the author, the study of usages of smartphone applications by individual users has been rather limited in the literature. This is so largely because of the extreme difficulty for acquiring real data needed for the study. Indeed, the existing literature has been relying upon real data collected from either a limited number of volunteers or the network level information at the sacrifice of details of individual users. In addition, financial performance measures of such digital products services are hard to come by. Recently, this seemingly impossible task of gathering information about installments and un-installments of smartphone applications from individual devices has been overcome by a Japanese software development company named Fuller, Inc. by establishing a win-win relationship with Android smartphone users.

Smartphones typically have the very short battery-life because of too many idle applications consuming a substantial amount of energy. By recognizing this essential pitfall of smartphones, in November 2012, Fuller, Inc., introduced a software package called “Mr. Mobile, the battery saver” which enables the company to acquire the information about which smartphone applications are downloaded to and removed from the devices that are registered with agreement of customers. In exchange, the customers receive the free assessment of the level of energy consumption for each smartphone application in their device along with recommendations concerning which idle applications should be removed.

In this thesis, the large volume of data acquired from Fuller, Inc. together with

the ranking information provided by Google in place of financial performance measures would be utilized to analyze mobile applications in detail.

1.2 Literature Review

In this subsection, a succinct summary of the literature relevant to this thesis is presented. Previous papers studying the usage of smartphone applications have been relying upon a limited set of usage data collected from a small number of volunteers. In (Falaki, Mahajan, Kandula, Lymberopoulos, Govindan, & Estrin, 2010), for example, real data were collected from 255 smartphone users from the two different smartphone platforms for periods of 7 to 28 weeks. In (Minh, Do, Blom, & Gatica-Perez, 2011), another analysis was carried out so as to understand the application usage in the contexts of location and proximity based on real data collected from 77 smartphone users for a period of 9 months. A mechanism called "Prediction Algorithm with Fixed Cycle Length (PAFCL)" was proposed in (Chang, Qi, Enhong, & Hui, 2012) for predicting mobile user's application usages, which was tested against real data collected from 38 volunteers.

A different approach for collecting real data concerning usages of smartphone applications is to rely upon the network level information at the sacrifice of details of individual users. In (Xu, et. al. 2011), for example, based on IP level network data from a cellular network in US for one week, analyzed were diurnal patterns of different application types, network access patterns and their relations with the location. Using the information of mobile in-app advertisements in network traces, (Tongaonkar, Dai,

Nucci, & Song, 2013) introduced a method of understanding mobile application usage patterns in the Android environment.

These papers discussed above are limited in that the focus is on aspects of energy or resource consumption of a device on daily basis over a short time period. To the best knowledge of the author, competitive performance measures of smartphone applications have been hardly studied. Some exceptions include: (Falaki, Mahajan, Kandula, Lymberopoulos, Govindan, & Estrin, 2010) where a relative popularity of smartphone application was measured in terms of the interaction time with the application, number of sessions and session length; (Mizusawa, Sumita, & Takano, 2015) in which the competitive structure of the smartphone game applications was examined; (Perera, Dewi, Sari, & Sumita, 2014), where application characteristics such as popularity, stability and potential risk were investigated based on the user reviews, application permission data and usage data for a period of 6 months; and (Perera, Shigeno, Sumita, & Yamamoto, 2016) which established a novel statistical approach for estimating the values of key competitive performance measures with respect to popularity and stability of mobile applications.

Through the discussions above, one finds that performance characteristics of mobile applications have not been studied in depth based on real data collected from a substantial number, say hundreds of thousands, of individual users over a significant length, say over a year. In order to fill this gap, this thesis addresses itself to the problem of how to identify the characteristics of smartphone applications of "sustainable popularity ranking" based on massive real data provided by Fuller, Inc.,

together with the ranking information provided by Google in place of financial performance measures.

1.3 Purpose of This Thesis

This thesis is written based on the large volume of data acquired from Fuller, Inc., which contain daily usage information about which applications are present and/or removed over the period of July 2013 through April 2016 collected from over 2 million devices (with alpha-numeric ID only) having Android OS. Based on the massive data above together with the ranking information provided by Google, the purpose of this thesis is to develop segmentation algorithms for identifying "smartphone applications with sustainable popularity ranking." Formally, let $t_0(a)$ be the month in which application 'a' is introduced into the market. Then application 'a' is said to be with sustainable popularity ranking associated with (M, m_0, m_1) if it is ranked within top M by Google Play at least once during the first m_0 months since $t_0(a) + 1$, and is still ranked within top M at least once during the m_0 months after $t_0(a) + 1 + m_1$. The resulting segmentation algorithms are applied to mobile applications in the category of Japanese free games, demonstrating the effectiveness and usefulness of them.

For identifying smartphone applications with sustainable popularity ranking, the first step of our analysis is to develop segmentation algorithms using Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Neural Network (NNet) and Naïve Bayes Classifier (NBC) provided by the R language. More specifically, two new segmentation algorithms are constructed by

combining the results of the individual six methods. One approach for the integration of the individual results is based on the majority voting, and the other by taking the optimal linear combination of the individual results.

While the two segmentation algorithms show excellent performance for identifying those smartphone applications with sustainable popularity ranking, they may not specify the key factors behind the segmentation. In order to overcome this pitfall, we secondly focus on LR and DT and develop five different segmentation algorithms. This approach enables one to analyze some hidden characteristics of smartphone applications with sustainable popularity ranking by extracting key factors that play a significant role in the segmentation algorithms, thereby detecting the driving force behind the segmentation.

These segmentation algorithms are applied to mobile phone applications in the category of Japanese free games. Extensive numerical results reveal some subtle characteristics of those free games with sustainable popularity ranking, providing useful information for software development companies. Based on the daily acquisition of tens of thousands of dynamic installation / uninstallation information of individual users and the publication of difficult-to-obtain financial information acquired from Fuller Inc., the contribution of this thesis is to build a comprehensive support to application developers regarding sustainable or unsustainable application. From the actual market data on free games, we developed useful analytical methods for smartphone and application development companies and extracted market characteristics, opening up new horizons not seen in conventional research.

The structure of this thesis is as follows. In the current chapter, we discussed the background behind the study and a succinct summary of the literature relevant to the work was presented. The purpose of this thesis was then described. In Chapter 2, the data set acquired from Fuller Inc. is discussed in detail. The concept of smartphone applications with sustainable popularity ranking is then introduced, along with the definition of an application profile vector for application 'a', denoted by $APF(a)$, which captures the usage pattern of application 'a'. We also describe Learning Data and Testing Data that will be needed for segmentation of sustainable popularity ranking.

Chapter 3 reviews, from the literature, the prevalent segmentation algorithms to be used for this thesis. We first discuss the essential structure of segmentation algorithm and then describe a general procedure for constructing a segmentation algorithm based on Learning Data and Testing Data. Furthermore, the concept of confusion matrix is described so as to define performance measures of a segmentation algorithm. The six prevalent segmentation algorithms, that are LR, DT, RF, SVM, NBC and NNet, are then explained in a succinct manner.

In Chapter 4, two new segmentation algorithms are constructed for identifying mobile applications with sustainable popularity ranking, where the results of the six individual methods above are integrated through two different approaches: one approach based on the majority voting and the other approach by taking the optimal linear combination of the individual results. Numerical results are presented by applying the two segmentation algorithms to Japanese free games, demonstrating speed and accuracy of them.

Chapter 5 addresses itself with the challenging problem of how to detect the driving force behind the segmentation. By focusing on LR and DT, the optimal threshold is determined through five different approaches, resulting in five different segmentation algorithms. These segmentation algorithms are applied to Japanese free games. Numerical results reveal that several components of the application profile vector play a key role behind the segmentation algorithms, thereby providing subtle and useful implications for application developers. Finally, in Chapter 6, some concluding remarks are given, indicating the future direction of the study.

Chapter 2

Smartphone Applications with Sustainable Popularity Ranking

2.1 Data Description

In this thesis, we employ data acquired from Fuller Inc. concerning Android smartphone applications, which contain Device ID, Application ID and the number of installments, activations and un-installments of individual applications in 25 different categories among others over a period of 34 months from July 2013 to April 2016. In order to represent the performance of individual smartphone applications, we adopt the daily ranking information provided by Google. The data structure is exhibited in Table 2.1, where the first column corresponds to Device ID and the second column contains Application ID. For the third column on, '1' in Month_k indicates that the application in the row was present in the device in the same row in Month_k, while '0' in Month_k means that the application was not present in the device throughout Month_k. Further details are given in Data Set 1 in Figure 2.1.

Device ID	Application ID	Month_01	…	Month_k	…	Month_21
50b7518eb775bc3ec	jp.gungho.pad	1	…	1	…	1
50b7518eb775bc3ec	jp.naver.SJLINEPANG	0	…	0	…	0
…	…	…	…	…	…	…

Table 2.1 Monthly Usage Information over the 21 Month Period

Also of interest is the ranking information obtained from Google Play, exhibited in Data Set 2 and Data Set 3 in Figure 2.1 below, where Android smartphone applications are listed with such information as Application ID, Application Name, Category, Price, Application Size, Developer, Daily Ranking and the like. In this thesis, the three sets of data are used together, where they are linked through the key Application ID.

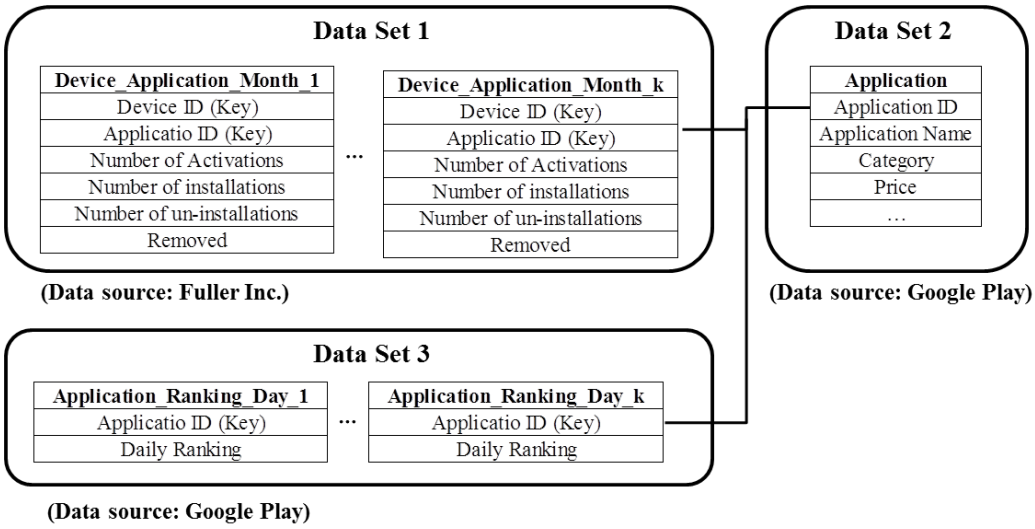


Figure 2.1 Data Structure

For the data set described above, the following notation would be employed.

1) $T = \{0, .., \mathrm{T}\}$: the period covered by the dataset (7/2013 to 4/ 2016)
2) $D(T)$: the set of devices registered in T
3) $A(T)$: the set of mobile applications under consideration, which are installed at least once in T in some $d \in \mathrm{D}(T)$
4) $I(a,d,t) = \begin{cases} 1 & \text{if } a \in \mathrm{A}(T) \text{ is present in } d \in D(T) \text{ in month } t \in T \\ 0 & \text{else} \end{cases}$
5) $t_0(a) = \min\{t: \sum_{d \in D(T)} I(a,d,t) > 0\}$: the month in which $a \in A(T)$ appeared first
6) $FA(t) = \{a \in A(\mathrm{T}): t_0(a) = t\}$: the set of applications that appear for the first time in month t $\in T \setminus \{0\}$

7) $Rank\ (a,t)$: ranking of $a \in A(T)$ in month $t \in T$

We note in 6) that $FA(t)$ can be constructed by picking up those applications that have not been present in any $d \in D(T)$ until t, and therefore the first month $\{0\}$ is excluded. The monthly ranking in 7) is obtained by taking the average of daily rankings in the month.

2.2 Concept of Sustainable Popularity Ranking

The concept of the sustainable popularity ranking is newly introduced now so as to identify those smartphone applications that become popular within a short time period after its release and remain popular even after a certain time period. Of interest then is to develop segmentation algorithms for separating smartphone applications with sustainable popularity ranking from others. More formally, a smartphone application $a \in A(T)$ is said to be with sustainable popularity ranking associated with (M, m_0, m_1) if it is ranked within top M by Google Play at least once during the first m_0 months since $t_0(a) + 1$, and is still ranked within top M at least once during the m_0 months after $t_0(a) + 1 + m_1$. The former interval is denoted by

$$BEG(t_0, m_0) = \{t_0 + 1, \cdots, t_0 + m_0\} \tag{2.2.1}$$

and the latter interval by

$$END(t_0, m_0, m_1) = \{t_0 + m_1, \cdots, t_0 + m_1 + m_0\}\ . \tag{2.2.2}$$

The definition of sustainable popularity ranking associated with (M, m_0, m_1) is illustrated in Figure 2.2.

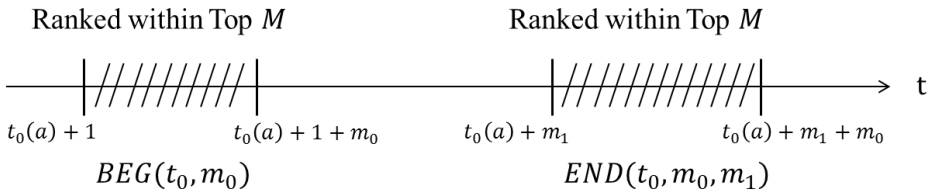


Figure 2.2 Sustainable Popularity Ranking Associated with (M, m_0, m_1)

Let two indicator functions be defined as

$$I_{Beg}(a, M, m_0) = \begin{cases} 1 & \text{if } Rank\ (a,t) \leq \text{M for } t\ \in BEG(t_0, m_0) \\ 0 & \text{else} \end{cases}, \quad (2.2.3)$$

and

$$I_{End}(a, M, m_0, m_1) = \begin{cases} 1 & \text{if } Rank\ (a,t) \leq \text{M for } t\ \in END(t_0, m_0, m_1) \\ 0 & \text{else} \end{cases}. \quad (2.2.4)$$

Then, the condition for $a \in A(T)$ to be with sustainable popularity ranking associated with (M, m_0, m_1) can be written as $I_{Beg}(a, M, m_0) = 1$ and $I_{End}(a, M, m_0, m_1) = 1$.

2.3 Application Profile Vector

For the purpose of developing segmentation algorithms for identifying smartphone applications with sustainable popularity ranking, it is necessary to describe each $a \in A(T)$ in terms of a set of variables. A vector consisting of such variables is called a profile vector of $a \in A(T)$ denoted by $APF(a)$. In order to construct $APF(a)$, we define two variables

$$sex \in \{male, female\} \quad (2.3.1)$$

and

$$age \in \{10, early\ 20,\ late\ 20,\ 30, 40, +50\}\ . \quad (2.3.2)$$

Let $D_{sex,age}(T)$ be the set of devices in $D(T)$ which are owned by those with $sex \in \{male, female\}$ and $age \in \{10, early\ 20,\ late\ 20, 30, 40, +50\}$. The application profile vector $APF(a)$ is then defined to be composed of the following eight different types of variables.

1) $Act\text{-}Count(a, \text{sex}, \text{age}, t)$: the sum of the number of activations of $a \in A(T)$ in month t over $d \in D_{sex,age}(T)$ (2.3.3)

2) $Ins\text{-}Count(a, \text{sex}, \text{age}, t)$: the sum of the number of installments of $a \in A(T)$ in month t over $d \in D_{sex,age}(T)$ (2.3.4)

3) $Unins\text{-}Count(a, \text{sex}, \text{age}, t)$: the sum of the number of un-installments of $a \in A(T)$ in month t over $d \in D_{sex,age}(T)$ (2.3.5)

4) $Possess\text{-}Count(a, \text{sex}, \text{age}, t)$: the number of devices in $D_{sex,age}(T)$ which possess $a \in A(T)$ in month t (2.3.6)

5) $Act\text{-}Count\text{-}Change(a, \text{sex}, \text{age}, l_0, l_1) = \frac{Act-Count(a,\text{sex},\text{age},l_1)}{\max\{Act-Count(a,\text{sex},\text{age},l_0),\ 1\}}$ (2.3.7)

6) $Ins\text{-}Count\text{-}Change(a, \text{sex}, \text{age}, l_0, l_1) = \frac{Ins-Count(a,\text{sex},\text{age},l_1)}{\max\{Ins-Count(a,\text{sex},\text{age},l_0),\ 1\}}$ (2.3.8)

7) $Unins\text{-}Count\text{-}Change(a, \text{sex}, \text{age}, l_0, l_1) = \frac{Unins-Count(a,\text{sex},\text{age},l_1)}{\max\{Unins-Count(a,\text{sex},\text{age},l_0),\ 1\}}$ (2.3.9)

8) $Possess\text{-}Count\text{-}Change(a, \text{sex}, \text{age}, l_0, l_1) = \frac{Possess-Count(a,\text{sex},\text{age},l_1)}{\max\{Possess-Count(a,\text{sex},\text{age},l_0),\ 1\}}$ (2.3.10)

We note that $Act\text{-}Count\text{-}Change(a, \text{sex}, \text{age}, l_0, l_1)$ in (2.3.7) describes the change of the number of activations of $a \in A(T)$ over $d \in D_{sex,age}(T)$ between the month l_0 and l_1 expressed in terms of the ratio of the two, where $\max\{Act - Count(a, \text{sex}, \text{age}, l_0),\ 1\}$ in the denominator is to make the formula valid even when $Act - Count(a, \text{sex}, \text{age}, l_0) = 0$. The formulas in (2.3.7) through (2.3.10) are defined similarly.

Segmentation algorithms can now be developed for identifying smartphone applications with sustainable popularity ranking associated with (M, m_0, m_1). More specifically, let

$$A_{Beg}(M, m_0, t) = \{a : a \in FA(t) \; and \; I_{Beg}(a, M, m_0) = 1\} \quad ; \qquad (2.3.11)$$

$$A_{End}(M, m_0, m_1, t) = \{a : a \in FA(t) \; and \; I_{End}(a, M, m_0, m_1) = 1\} \quad . \; (2.3.12)$$

Then, our task is to identify $a \in A_{Beg}(M, m_0, t)$ which also satisfies $a \in A_{End}(M, m_0, m_1, t)$.

In Chapter 3, six prevalent segmentations algorithms are reviewed, which are used, in Chapters 4 and 5, to establish a variety of segmentation algorithms by combining the results of multiple segmentation algorithms so as to enhance the performances of the individual segmentation algorithms.

2.4 Learning Data and Testing Data for Identifying Mobile Applications with Sustainable Popularity Ranking

For the data acquired from Fuller Inc., we now design Learning Data and Testing Data for developing segmentation algorithms to identify mobile applications with popularity ranking.

For l_0 and l_1 in (2.3.7) through (2.3.10), we set $l_1 - l_0 = 2$, $m_0 = 2$ and $m_1 = 6$. This means that we are interested in those smartphone applications that became popular within 3 months since its release and will be still popular after 6 months. The changes of activations, installments and un-installments are measured every 3 months. The values of l_1, l_0, m_0 and m_1 can be shifted accordingly to conduct a sensitivity analysis that is not addressed in this thesis.

For constructing the first Learning Data, one has $t_0 = t = 1 =$ August 2013, $BEG(t_0, m_0) = \{2, 3, 4\}$ and $END(t_0, m_0, m_1) = \{7, 8, 9\}$. A segmentation algorithm is then to attempt to identify $a \in A_{Beg}(M, m_0, 1)$ which also satisfies $a \in A_{Beg}(M, m_0, 1) \cap A_{End}(M, m_0, m_1, 1)$. So as to obtain enough Learning Data, we repeat the above structure 11 times more by shifting it by one month at a time. After this process, the subsequent combination of t_0, $BEG(t_0, m_0)$, and $END(t_0, m_0, m_1)$

would be used as Testing Data. This structure constitutes one period to examine the validity of a segmentation algorithm, which we denote by $\text{Period}\,(1) = \{1, \cdots, 22\}$.

More specifically, one has

$$\text{LD}(1) = \bigcup_{t=1}^{12} A_{Beg}(M, m_0, t) \tag{2.4.1}$$

and

$$\text{TD}(1) = A_{Beg}(M, m_0, 14)\ . \tag{2.4.2}$$

We also construct a segmentation algorithm out of the results of multiple segmentation algorithms. In this case,

$$\text{LD}_1(1) = A_{Beg}(M, m_0, 14) = \text{TD}(1) \tag{2.4.3}$$

is used as the second set of learning data to determine the optimal way of combining the individual results of the underlying segmentation algorithms obtained through (2.4.1). The data set for testing the validity of the combined segmentation algorithm is given by

$$\text{TD}_1(1) = A_{Beg}(M, m_0, 15)\ . \tag{2.4.5}$$

Figure 2.4.1 illustrates the Learning and Testing Data structure for the first period discussed above, where the blue cells are designated to indicate t_0, while the green cells represent $BEG(t_0, m_0)$, and the red cells correspond to $END(t_0, m_0, m_1)$. The vertical axis exhibits the learning-testing data structure, where the first 12 rows correspond to $A_{Beg}(M, m_0, t)$ and $A_{End}(M, m_0, m_1, t)$ for $t = 1, \cdots, 12$. Since one period requires 22 months of data while we have 34 months of data, it is possible to repeat the validity test of the segmentation algorithm 12 times, over Period (1) through Period (12) by shifting each period by one month at a time. In parallel with (2.4.1)

through (2.4.5), we define $\mathrm{LD}(\tau), \mathrm{LD}_1(\tau)$, $\mathrm{TD}(\tau)$, and $\mathrm{TD}_1(\tau)$ for $\tau = 2, \cdots, 12$ similarly.

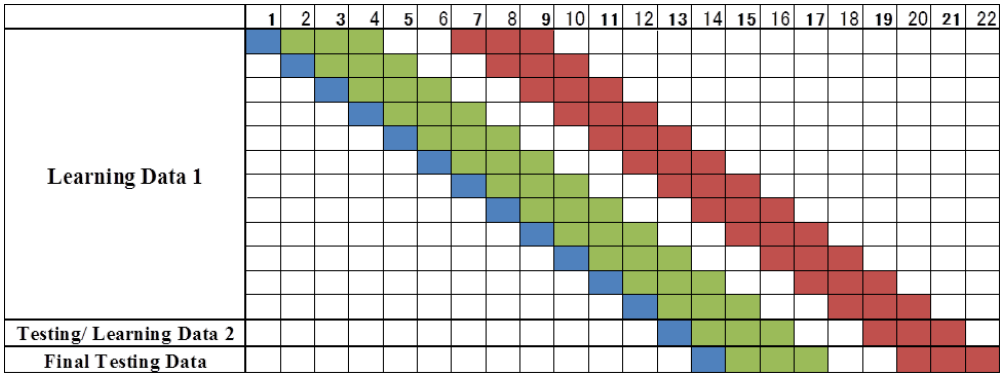


Figure 2.4.1 Design of Learning Data and Testing Data for Period (1)

Chapter 3

Prevalent Segmentation Algorithms

In this chapter, we provide a succinct summary of the concept of a segmentation algorithm, Learning Data, Testing Data, the confusion matrix, and traditional performance measures of segmentation algorithms. Furthermore, several prevalent segmentation algorithms are discussed, which will be employed in this thesis as a basis to construct our own new segmentation algorithms. Concerning these segmentation methods, the reader is referred to (Hosmer, Lemeshow, & Sturdivant, 2013; Buja, & Lee, 2001; Joachims, 2002; Jelinek, 2005; Piovoso, & Owens, 1991; Lin, Wang, & Zou, 2015) for further details. The reader is also referred to an excellent introductory book entitled "Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data" by EMC Educational Services (2015).

3.1 Essence of Segmentation Algorithm

We first consider a simple marketing case to understand the concept of segmentation algorithm. Let $CS = \{c_1, \cdots, c_N\}$ be a set of N customers under consideration. Associated with each customer c_i is the profile vector x_i, typically describing the basic information of c_i and his/her past purchasing behavior. According to a prespecified criterion, we suppose that a set of good customers will be determined by their purchasing outcome in the next future period. More specifically, one has $CS = G_{Next} \cup B_{Next}$, where G_{Next} and B_{Next} denote the set of good customers and

the set of bad customers in the next future period, respectively, which would be realized only upon completion of the next future period. At the present time, of interest is to develop a segmentation algorithm which would attempt to identify those customers in G_{Next} by estimating customers' future purchasing behavior based on $x_i,\ i = 1, \cdots, N$. In what follows, we formally describe this concept in a generic manner.

Let A be a set of data under consideration, where each element $a \in A$ has the profile vector $\underline{x}^T(a) = [x_1(a), \cdots x_N(a)]$ and a flag $Flag(a) \in \{0, 1\}$. Associated with $\underline{x}^T(a)$ is the flag function $I_{True}\colon\ A \to \{0, 1\}$. In the previous marketing example described above, $\underline{x}^T(a)$ would consist of the basic information of a and his/her purchasing records up to the previous month. The value of $I_{True}(a)$ would be either 0 or 1 and would be determined by the purchasing records of a in the current month, stating whether or not a was a good customer in the current month. For our mobile phone application model discussed in Chapter 2, $\underline{x}^T(a)$ is the application profile vector.

A segmentation algorithm can be represented by a mapping $I_{Seg}\colon\ A \to \{0, 1\}$, defined by

$$I_{Seg}(a) = \begin{cases} 1 & \text{if the segmentation algorithm judges } I_{True}(a) = 1 \\ 0 & otherwise \end{cases} . \qquad (3.1.1)$$

In other words, $I_{Seg}(a)$ estimates the future behavior of a, resulting in the segmentation of A into $A = A_{Seg}(1) \cup A_{Seg}(0)$, where

$$A_{Seg}(j) = \{a \in A\colon I_{Seg}(a) = j\},\ j = 0\ or\ 1\ . \qquad (3.1.2)$$

In what follows, we describe a general procedure for constructing I_{Seg}: $A \to \{0,1\}$.

3.2 Construction of Segmentation Algorithm through Learning and Testing Data

The first step to develop a segmentation algorithm is to decompose A into a set of learning data A_L and a set of testing data A_T, that is, $A = A_L \cup A_T$, $A_L \cap A_T = \emptyset$, $A_L \neq \emptyset, A_T \neq \emptyset$. In many cases, A_L and A_T are chosen randomly. A typical procedure would be to develop a segmentation algorithm $I_{Seg}(a|\underline{\beta}, pp)$ for each $a \in A_L$ involving a parameter vector $\underline{\beta}$ based on the profile vectors of the values up to the previous period (*pp*). Namely, $I_{Seg}(a|\underline{\beta}, pp)$ returns 0 or 1 based on the records up to the previous period so as to estimate $Flag(a|cp)$ in the current period (cp). The optimal parameter vector $\underline{\beta}^*$ is determined by applying $I_{Seg}(a|\underline{\beta}, pp)$ for $a \in A_L$ and then minimizing the distance to $Flag(a|pp)$ defined by the profile vectors of the values up to the previous period. More specifically, one has

$$\underline{\beta}^* = arg\min_{\underline{\beta}} \left[\sum_{a \in A_L} \left\{ I_{Seg}\left(a \middle| \underline{\beta}, pp\right) - Flag(a|pp) \right\}^2 \right] \quad . \tag{3.2.1}$$

The resulting segmentation algorithm $I_{Seg}(a|\underline{\beta}^*, pp)$ is applied for $a \in A_T$ to test the performance by comparing it with $Flag(a|cp)$. If it is acceptable, then $I_{Seg}(a|\underline{\beta}^*, pp)$ is used to estimate $I_{True}(a|np)$ during the next period (*np*) by applying it to the profile vectors of the values in the current period.

Some segmentation algorithms may take an intermediary function

$Seg: A \rightarrow [0,1]$, where $Seg(a)$ is a number between 0 and 1, rather than 0 or 1. In this case, one may decompose A into a set of learning data A_L and two sets of testing data A_{T1} and A_{T2}, that is, $A = A_L \cup A_{T1} \cup A_{T2}$, $A_L \cap A_{T1} = \emptyset$, $A_L \cap A_{T2} = \emptyset$, $A_{T1} \cap A_{T2} = \emptyset, A_L \neq \emptyset, A_{T1} \neq \emptyset, A_{T2} \neq \emptyset$. A segmentation algorithm $Seg(a|\underline{\beta})$ for each $a \in A_L$ involving a parameter vector $\underline{\beta}$ is first constructed based on the profile vectors of the values up to the previous period. In parallel with (3.2.1), the optimal parameter vector $\underline{\beta}^*$ is then determined by

$$\underline{\beta}^* = arg\min_{\underline{\beta}} \left[\sum_{a \in A_L} \left\{ Seg\left(a \middle| \underline{\beta}, pp\right) - Flag(a|pp) \right\}^2 \right] \quad . \tag{3.2.2}$$

The next step is to define $I_{Seg}(a|\underline{\beta}^*, z, pp)$ by

$$I_{Seg}(a|\underline{\beta}^*, z, pp) = \begin{cases} 1 & \text{if } Seg\left(a \middle| \underline{\beta}^*, pp\right) \geq z \\ 0 & otherwise \end{cases} \quad . \tag{3.2.3}$$

The threshold value z is optimized by applying $I_{Seg}(a|\underline{\beta}^*, z, pp)$ for $a \in A_{T1}$ by following the procedure to be discussed in the next section, yielding z^*. Finally, the performance of the segmentation algorithm $I_{Seg}(a|\underline{\beta}^*, z^*, pp)$ can be tested by applying it to $a \in A_{T2}$.

3.3 Confusion Matrix and Related Performance Measures

The validity of the segmentation algorithm can be examined by applying it to the set of Testing Data, where the following confusion matrix is constructed.

		$I_{True}(a)$		Sum
		0	1	
$I_{Seg}(a)$	0	x_{00}	x_{01}	X_0
	1	x_{10}	x_{11}	X_1
Sum		Y_0	Y_1	N

(3.3.1)

Here, x_{mn} denotes the number of data satisfying $I_{Seg}(a) = m$ and $I_{True}(a) = n$, $m, n = 0, 1$. One also has $X_i = x_{i0} + x_{i1}$, $Y_j = x_{0j} + x_{1j}$, $i, j = 0, 1$, and $N = X_0 + X_1 = Y_0 + Y_1$. For testing the validity of the segmentation algorithm, the three traditional indices are

$$Recall = \frac{x_{11}}{Y_1};\ Precision = \frac{x_{11}}{X_1};\ Accuracy = \frac{x_{00}+x_{11}}{N} \quad . \tag{3.3.2}$$

We note that *Recall* describes the portion of those a with $I_{Seg}(a) = 1$ over Y_1 (i.e. over those with $I_{True}(a) = 1$), while $Precision$ represents the portion of those applications a with $I_{True}(a) = 1$ over X_1 (i.e. over those with $I_{Seg}(a) = 1$). Similarly, $Accuracy$ is the index for the overall correctness of the segmentation algorithm.

It is known, see e.g. Mizuno, Saji, Sumita and Suzuki (2008) that $Recall$ and $Precision$ have a trade-off relationship in that one typically increases as the other

decreases. In order to achieve the balance between $Recall$ and $Precision$, the fourth index called *f-measuer* is defined as

$$f = \frac{1}{\frac{1}{2}\left(\frac{1}{Recall}+\frac{1}{Precision}\right)} \quad . \tag{3.3.3}$$

When the intermediary function $Seg(a|\underline{\beta}, pp)$ and the corresponding segmentation algorithm $I_{Seg}(a|\underline{\beta}^*, z, pp)$ is determined via (3.2.3), the confusion matrix in (3.3.1) as well as the four performance measures in (3.3.2) and (3.3.3) would depend on z. In this case, the optimal threshold z^* may be determined by solving

$$z^* = arg\max_{0\le z\le 1}[Precision(z)\ subject\ to\ Recall(z) \ge \alpha\ ] \quad . \tag{3.3.4}$$

The intent of this optimization problem is to maximize the probability of correct estimation subject to at least $100\times\alpha\ \%$ of those α with $I_{True}(a) = 1$ to be picked up by the segmentation algorithm.

In what follows, several prevalent segmentation algorithms are discussed, which will be employed in this thesis as a basis to construct our own new segmentation algorithms.

3.4 Logistic Regression (LR) Approach

The LR model is often employed for estimating the probability of whether or not a certain event occurs. Let D_L be Learning Data which consists of explanatory variable vectors of the form $\underline{x} = [1, x_1, \cdots, x_n]^T, \underline{x} \in D_L$, and let D_T be Testing Data

with $\underline{y} = [1, y_1, \cdots, y_n]^T, \underline{y} \in D_T$. It should be noted that the domain of D_L should be the same as that of D_T, which is denoted by Ω. Let $I(\underline{x})$ be the label of $\underline{x}$ taking the value of either 0 or 1 and define $I(\underline{y})$ similarly. The purpose of LR is to find a way of determining the probability of $I(\underline{y}) = 1$, denoted by $p(\underline{y})$, based on $\underline{x} \in D_L$ and $I(\underline{x})$. As we will see, this estimation is done through a form of regression. It is a common practice to standardize x_i and y_i so that $\Omega = [0,1]^N$. By doing this, one can better understand the weight of the role of individual variables in the estimation.

Let $\underline{\alpha} = [\alpha_0, \alpha_1, \cdots, \alpha_N]^T$ and define $w: \Omega \to R$ as

$$w(\underline{x}|\underline{\alpha}) = \underline{\alpha}^T \underline{x} = \alpha_0 + \sum_{i=1}^{N} \alpha_i x_i \quad . \tag{3.4.1}$$

The desired probability given $\underline{\alpha}^T$, denoted by $p\left(\underline{y}|\underline{\alpha}\right)$, is then estimated by

$$log\left(\frac{p\left(\underline{y}|\underline{\alpha}\right)}{1-p\left(\underline{y}|\underline{\alpha}\right)}\right) = w(\underline{x}|\underline{\alpha}) \quad . \tag{3.4.2}$$

By solving (3.4.2) for $p\left(\underline{y}|\underline{\alpha}\right)$, one finds that

$$p\left(\underline{y}|\underline{\alpha}\right) = \frac{e^{w(\underline{x}|\underline{\alpha})}}{1+e^{w(\underline{x}|\underline{\alpha})}} = \frac{1}{1+e^{-w(\underline{x}|\underline{\alpha})}} \quad . \tag{3.4.3}$$

The optimal coefficient vector $\underline{\alpha^*} = [\alpha^*{}_0, \alpha^*{}_1, \cdots, \alpha^*{}_N]^T$ can then be found by solving

$$\underline{\alpha}^* = argmin_{\underline{\alpha}} \left\{ \sum_{\underline{x} \in D_L} \left[I(\underline{x}) - p\left(\underline{y}|\underline{\alpha}\right) \right]^2 \right\} \tag{3.4.4}$$

Given a threshold z, the final segmentation judgment is then given by

$$I_{Seg}(\underline{y}|\underline{\alpha}^*, z) = \begin{cases} 1 & \text{if } p\left(\underline{y}|\underline{\alpha}^*\right) \geq z \\ 0 & otherwise \end{cases} . \tag{3.4.5}$$

3.5 Decision Tree (DT) Approach and Random Forest (RF) Approach

Decision Tree is a segmentation algorithm, which enables one to classify the underlying entities into a set of classes without relying upon a specific functional structure. Each entity is represented by a set of explanatory variables constituting an N dimensional vector. A source node specifies a branching condition, whereas a leaf node establishes a subclass of its source node. Since the branching processes can be traced explicitly with corresponding branching conditions, the reasoning behind the separation into the leaf classes can be interpreted easily.

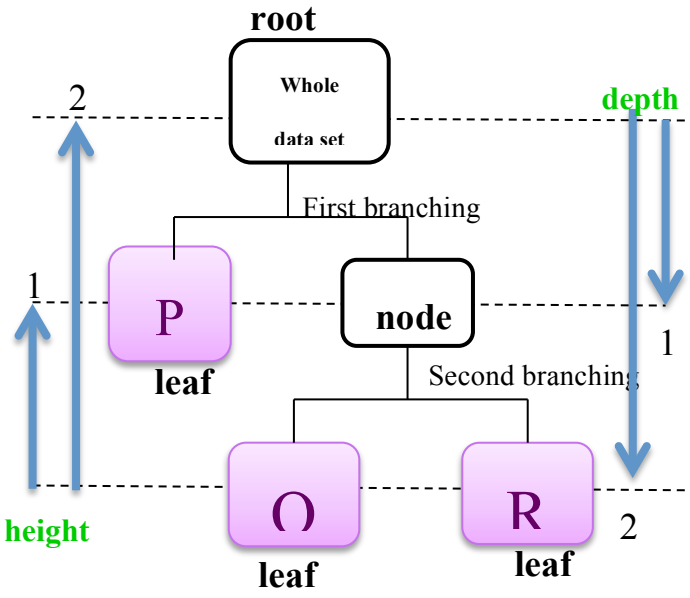


Figure 3.5.1 Basic Features of DT

The formal description of Decision Tree is as follows. Let Ω be a universal set of subjects under consideration. Let $\Omega = \cup_{m=1}^{M} A_j$, satisfying $A_m \neq \emptyset$, $A_m \cap A_n = \emptyset$ $(m \neq n)$. Each $i \in \Omega$ is characterized by its profile vector $\underline{x}(i) = [x(i,1), \cdots, x(i,K)] \in V$, and the class index vector $\underline{c}(i) = [c(i,1), \cdots, c(i,M)]$, where $c(i,m) = 1$ if $i \in A_m$ and $c(i,m) = 0$, otherwise. With $\mathbb{N} = \{1, \cdots, N\} \subset \Omega$, a Learning Data is defined as $LD = \{(\underline{x}(i), \underline{c}(i)) : i \in \mathbb{N}\}$. The problem is to establish an algorithm to estimate $\underline{c}(j)$ for each $j \in \Omega \setminus \mathbb{N}$, given $\underline{x}(j)$.

Let PN be a subset of Ω constituting a parent node, where the initial parent node is set to be $\mathbb{N}$. Let v be a branching condition, that is, the branching index function $\mathrm{I}(\underline{x}, v)$ is well defined for any $\underline{x} \in V$, where $I(\underline{x}, v) = 1$ if $\underline{x}$ satisfies v and $I(\underline{x}, v) = 0$ otherwise. Typically, the branching condition is based on the entropy, maximizing the difference between the sum of the entropies resulting from the branching and the entropy of the parent node.

Let $V(PN)$ be the set of branching decisions that have not been adopted before reaching PN. Then the algorithm can be described as below.

Given $B \subset \Omega$, let $P(m|B) = |B \cap A_m|/|B|$.

[1] Given PN, choose $v \in V(PN)$ and apply it to PN, yielding

$$PN_{Yes}(v) = \{i \in PN : I(\underline{x}(i), v) = 1\} \text{ and } PN_{No}(v) = PN \setminus PN_{Yes}(v).$$

[2] Compute

$$I_E(PN) = -\sum_{m=1}^{M} P(m|PN) \log_2 P(m|PN),$$

$$I_E(PN_{Yes}(v)) = -\sum_{m=1}^{M} P(m|PN_{Yes}(v)) \log_2 P(m|PN_{Yes}(v)),$$

and

$$I_E\big(PN_{No}(v)\big) = -\sum_{m=1}^{M} P\big(m|PN_{No}(v)\big)\log_2 P\big(m|PN_{No}(v)\big) \ .$$

[3] Compute $I_G\big(PN, PN_{Yes}(v), PN_{No}(v)\big) = I_E(PN) - I_E\big(PN_{Yes}(v), PN_{No}(v)\big)$.

[4] Find $v^* = \text{argmin}_{v\in V(PN)}\big\{I_G\big(PN, PN_{Yes}(v), PN_{No}(v)\big)\big\}$ by repeating [1] through [3].

[5] Set both $PN_{Yes}(v^*)$ and $PN_{No}(v^*)$ as new parent nodes.

[6] Repeat [1] through [5] until the entire tree is completed.

Random forest approach is designed to enhance the performance of DT by performing randomly chosen sub-DTs repeatedly and then integrating the results to form a final segmentation algorithm. More formally, let A be a set of data under consideration, where each element $a \in A$ has the profile vector $\underline{x}^T(a) = [x_1(a), \cdots x_N(a)]$. For $V_{Sub} \subset \{1, \cdots, N\}$, define $\underline{x}^T_{V_{Suv}}(a) = [x_i(a)]_{i\in V_{Sub}}$. Then the random forest approach can be describes as follows.

< Random Forest>

[1] Randomly choose a subset $A_{Sub} \subset A$ and a subset $V_{Sub} \subset \{1, \cdots, N\}$.

[2] Perform DT using $\underline{x}^T_{V_{Suv}}(a)$ for $a \in A_{Sub}$.

[3] Repeat [1] and [2] K times, yielding K outputs $Res_j(a), j = 1, \cdots K$..

[4] Integrate $Res_j(a), j = 1, \cdots K$ to produce $Res_{Int}(a)$, e.g. if $Res_j(a) \in \{0, 1\}$, take the majority vote, or if $Res_j(a) \in [0, 1]$, take the average, maximum or minimum, etc.

[5] Use $Res_{Int}(a)$ to construct a final segmentation algorithm.

3.6 Support Vector Machine (SVM) Approach

Given n vectors in R^N with each having the flag of value either 0 or 1, the goal of SVM is to establish a functional structure that separates the vectors with flag = 1 from those with flag = 0. Given a new vector, this functional structure enables one to judge the flag value of the new vector. One advantage of SVM over other segmentation methods can be found in that its accuracy does not deteriorate much as the dimension N increases. However, it is rather a difficult task to decide which explanatory variables would influence the value of the flag.

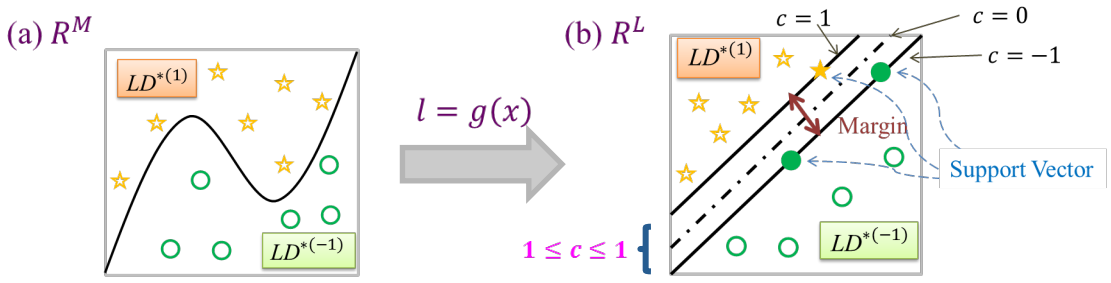


Figure 3.6.1 SVM (Support Vector Machine)

Let LD be a set of Learning Data defined by:

$$LD = \{(\underline{x}_n, Flag(\underline{x}_n)) : \underline{x}_n \in R^M, Flag(\underline{x}_n) = 0 \text{ or } 1, n = 1,2,\cdots,N\}.$$

The purpose of SVM is to establish a functional structure to separate $LD^{(1)} = \{(\underline{x}, 1)\}$ from $LD^{(-1)} = \{(\underline{x}, -1)\}$. Since such a functional structure is typically non-linear, a function $g: R^M \to R^L$ is introduced so that

$$LD^{*(1)} = \{\underline{g}(\underline{x}), 1\} \tag{3.6.1}$$

and

$$LD^{*(-1)} = \{\underline{g}(\underline{x}), 0\} \tag{3.6.2}$$

can be mostly separated by a separating hyperplane given by

$$D(\underline{x}) = \underline{w}^T \underline{g}(\underline{x}) + b \quad . \tag{3.6.3}$$

Hence the problem is now to determine $\underline{w}^T$ and b so that $\underline{x} \in LD^{(1)} \Rightarrow \underline{g}(\underline{x}) \in LD^{*(1)}$ and $\underline{x} \in LD^{(-1)} \Rightarrow \underline{g}(\underline{x}) \in LD^{*(-1)}$.

Those vectors satisfying $D(\underline{x}) = \underline{w}^T \underline{g}(\underline{x}) = c$ for $c = -1, 1$ are called support vectors of the minus side and the plus side, separating $LD^{(-1)} = \{(\underline{x}, -1)\}$ and $LD^{(1)} = \{(\underline{x}, 1)\}$ as can be seen in Figure 3.6.1. The problem is to find $\underline{w}^T$ which would maximize the distance between the two hyper planes $\underline{w}^T \underline{g}(\underline{x}) = -1$ and $\underline{w}^T \underline{g}(\underline{x}) = 1$. More formally, this problem can be formulated as:

$$min\ Q(\underline{w}) = \frac{1}{2} \|\underline{w}\|^2$$

$$s.t.\ \ y_i\left(\underline{w}^t\underline{g}(\underline{x}_i)+b\right)\geq 1 \qquad i\ =\ 1,\cdots,N \quad .$$

3.7 Naive Bayes Classifier (NBC) Approach

Let A be a set of data under consideration, where each element $a \in A$ has the profile vector $\underline{x}^T(a) = [x_1(a), \cdots x_N(a)]$ and a flag $Flag(a) \in \{0,1\}$. As discussed before, A is decomposed into a set of Learning Data A_L and a set of Testing Data A_T, that is, $A = A_L \cup A_T$, $A_L \cap A_T = \emptyset$, $A_L \neq \emptyset, A_T \neq \emptyset$. We assume that $x_i(a)$ is a realization of a random variable X_i, $i = 1, \cdots, N$. Let $A(j) = \{a \in A: Flag(a) = j\}$, $j = 0, 1$. Given $b \in A_T$, under the assumption that X_i, $i = 1, \cdots, N$ are mutually independent, Naive Bayes Classifier (NBC) is to determine whether or not $b \in A(1)$ based upon Bayes theorem applied to $\underline{x}^T(b)$ and $\underline{x}^T(a)$ for $a \in L$.

More specifically, we consider

$$P[\, b \in A(j) \mid X_1 = x_1(b), \cdots, X_N = x_N(b)\ ] = P\big[\, b \in A(j) \mid \underline{X} = \underline{x}^T(b)\,\big],\ j = 0, 1, \quad (3.7.1)$$

and choose j by comparing these probabilities, that is, we define

$$I_{Seg}(a) = \begin{cases} 1 & \text{if } P\big[\, b \in A(1) \mid \underline{X} = \underline{x}^T(b)\,\big] > P\big[\, b \in A(0) \mid \underline{X} = \underline{x}^T(b)\,\big] \\ 0 & otherwise \end{cases} \quad . \quad (3.7.2)$$

It remains to find $P\big[\, b \in A(j) \mid \underline{X} = \underline{x}^T(b)\,\big],\ j = 0, 1.$

From Bayes theorem, the probability $P\big[\, b \in A(j) \mid \underline{X} = \underline{x}^T(b)\,\big]$ in (3.7.1) can be rewritten as

$$P\left[\, b \in A(j) \mid \underline{X} = \underline{x}^T(b) \,\right] = P\left[\, b \in A(j)\,,\ \underline{X} = \underline{x}^T(b) \,\right] / P\left[\, \underline{X} = \underline{x}^T(b) \,\right]$$

$$= P[\, b \in A(j) \,] \cdot P\left[\underline{X} = \underline{x}^T(b) | b \in A(j)\right] / P\left[\, \underline{X} = \underline{x}^T(b) \,\right]. \qquad (3.7.3)$$

The left hand side $P\left[\, b \in A(j) \mid \underline{X} = \underline{x}^T(b) \,\right]$ is the posterior probability that we would like to evaluate. The denominator $P\left[\, \underline{X} = \underline{x}^T(b) \,\right]$ of the right hand side is called the evidence probability and would not affect the decision based on the comparison in (3.7.2), and therefore can be ignored for our purpose. The prior probability $P[\, b \in A(j) \,]$ may be simply estimated as $|A(j)|/|A|$, where $|S|$ denotes the cardinality of a set S. The likelihood probability $P\left[\underline{X} = \underline{x}^T(b) | b \in A(j)\right]$ can be estimated from $\underline{x}^T(a)$ for $a \in L$ in the following manner. From the assumption that $X_i,\ i = 1, \cdots, N$ are mutually independent, one sees that

$$P\left[\underline{X} = \underline{x}^T(b) | b \in A(j)\right]$$
$$= P[\, X_1 = x_1(b) \mid b \in A(j)] \cdot P[\, X_2 = x_2(b), \cdots, X_N = x_N(b) \mid b \in A(j),\, X_1 = x_1(b)]$$
$$= P[\, X_1 = x_1(b) \mid b \in A(j)] \cdot P[\, X_2 = x_2(b), \cdots, X_N = x_N(b) \mid b \in A(j) \,] \quad .$$

By repeating this procedure, it can be seen that

$$P\left[\underline{X} = \underline{x}^T(b) | b \in A(j)\right] = \prod_{i=1}^{N} P[\, X_i = x_i(b) \mid b \in A(j)] \quad . \qquad (3.7.4)$$

The probabilities $P[\, X_i = x_i(b) \mid b \in A(j)]$ can be estimated by evaluating the histogram of $\underline{x}^T(a)$ for $a \in L$, and therefore $P\left[\, b \in A(j) \mid \underline{X} = \underline{x}^T(b) \,\right]$ can be

estimated from (3.7.3) and (3.7.4).

3.8 Neural Network (NNet) Approach

A neural network is constructed by mimicking the signal-processing network structure of human brain. We first provide a succinct summary of how human brain works.

3.8.1 Neuron

Neurons transmit impulses as electric signal. These signals pass along the cell surface membrane of the axon as a nerve impulse. The speed of nerve impulses ranges from approximately 1 m/s to 120 m/s, which is much slower than an electric current passing down a wire. At rest, there are more negative ions inside the neuron compared with the outside with difference of about 70 millivolt. When a point on the semipermeable neural membrane is stimulated by an incoming message strongly enough to cause the change of 15~20 millivolt, the membrane opens at that point, and positively charged ions flow in. This process is repeated along the length of the membrane, creating the neural impulse of about 100 millivolt for 1 millisecond that travels down the axon, causing the neuron to fire. There are neurons of excitement type and those of suppression type, see Figure 3.8.1.1.

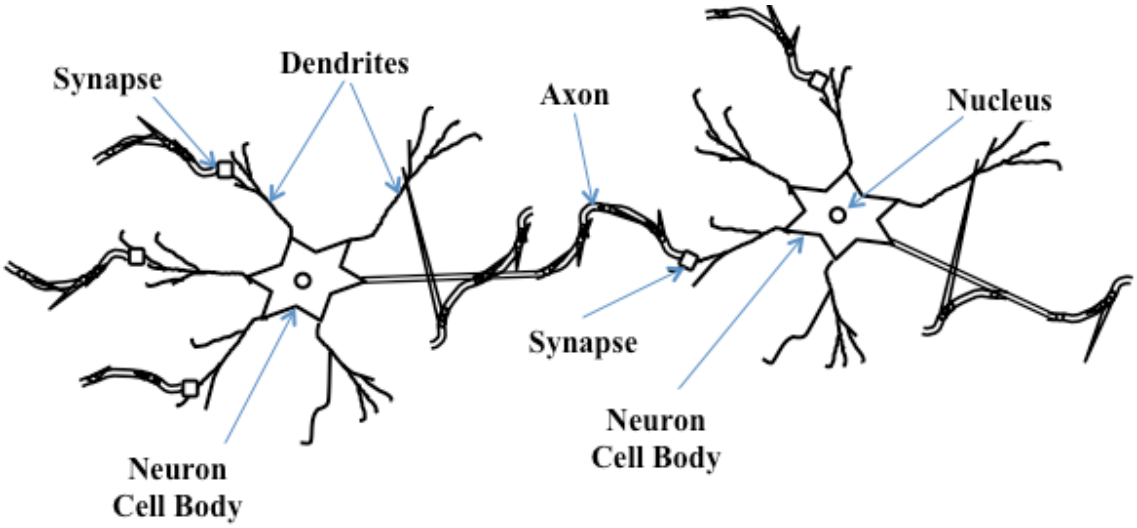


Figure 3.8.1.1 Generation of Nerve Impulse

3.8.2 Logical Operations Based on Perceptron

Perceptron is the generic name given by a Psychologist named Franck Rosenblatt to a family of theoretical and experimental artificial neural net models, which he proposed in the period 1957 – 1962. These signals pass along the cell surface membrane of the axon as a nerve impulse. By taking N inputs $x_i = 0\ or\ 1$ and weights $w_i, i = 1, \cdots, N,$ it produces the inner product $\underline{w}^T \underline{x} = \sum_{i=1}^{N} w_i\, x_i$. If x_i comes from a neuron of excitement type, one has $w_i > 0,$ while $w_i < 0$ if it comes from a neuron of suppression type. Given a threshold $\theta,$ the perceptron produces the output signal of 1 if $\underline{w}^T \underline{x} \geq \theta$. Otherwise, the output signal is 0. This process is depicted in Figure 3.8.2.1.

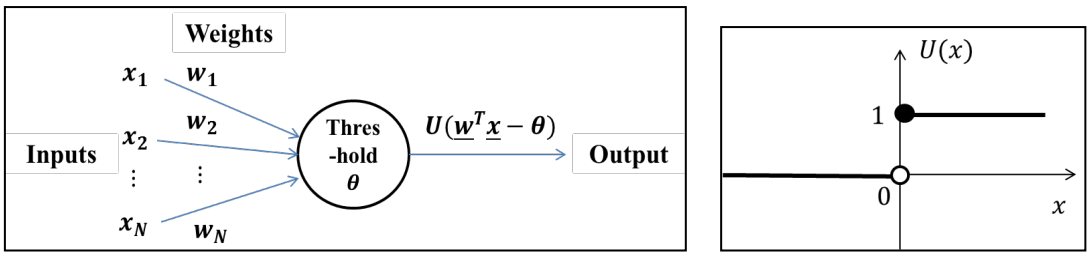
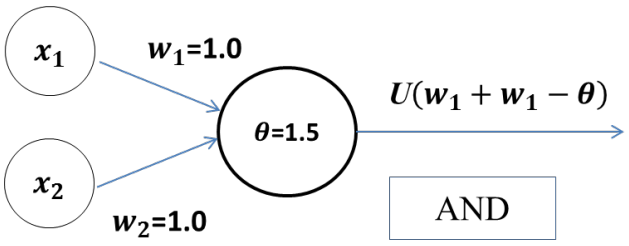
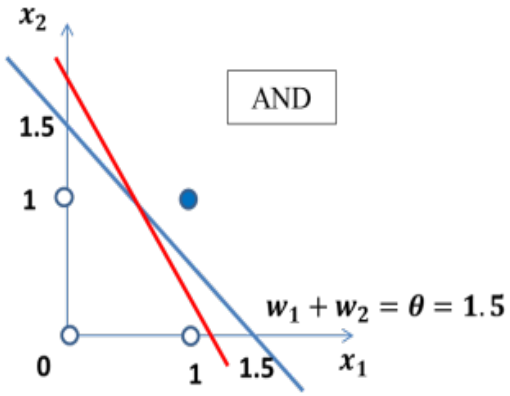


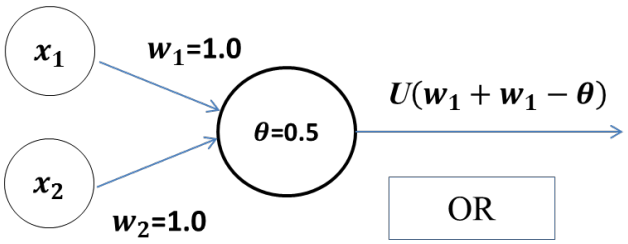
Figure 3.8.2.1 Structure of Perceptron and Pulse Function *U(x)*

The following example demonstrates how logical operations ∧ and ∨ via perceptron.

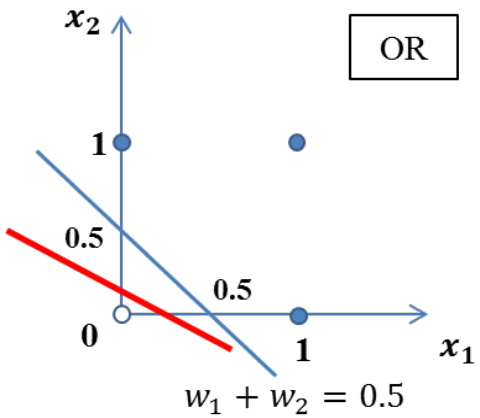


AND		x_2	
		0	1
x_1	0	0	0
	1	0	1





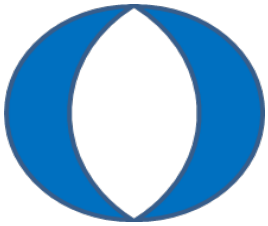
OR		x_2	
		0	1
x_1	0	0	1
	1	1	1

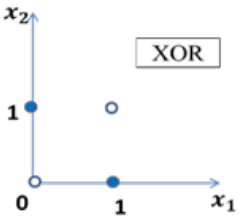


3.8.3 Limitation of Perceptron and Multi-layered Neural Network

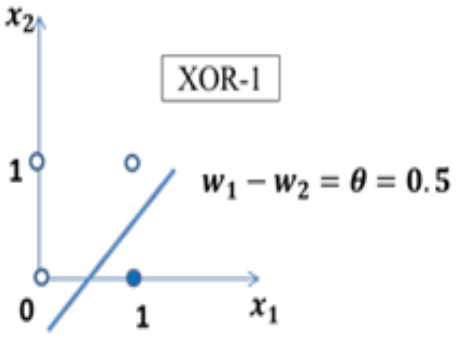
Logical operation via perceptron relies upon linear separation using the inner product $\underline{w}^T\underline{x}$ and a threshold θ. This approach has its limitation, as can be seen from the following example of the logical operation XOR which cannot be represented by any linear separation. We note that, in the third graph, the two black points cannot be separated from two white points by any single line.

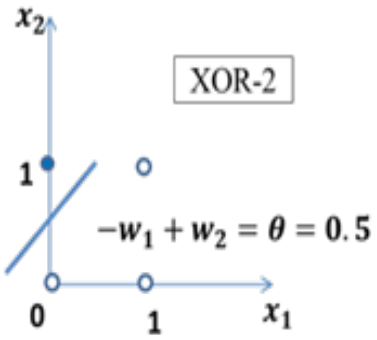
XOR		x_2	
		0	1
x_1	0	0	1
	1	1	0

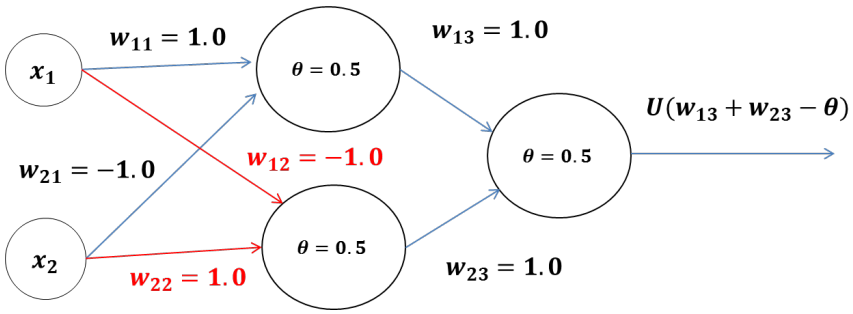




In order to separate the two black points from the two white points, the output of XOR-1 may be combined with that of XOR-2 through OR, as depicted below.







The above example may be called a two-layered perceptron. This leads to a more general approach based on a multi-layered neural network, as illustrated in Figure 3.8.3.1

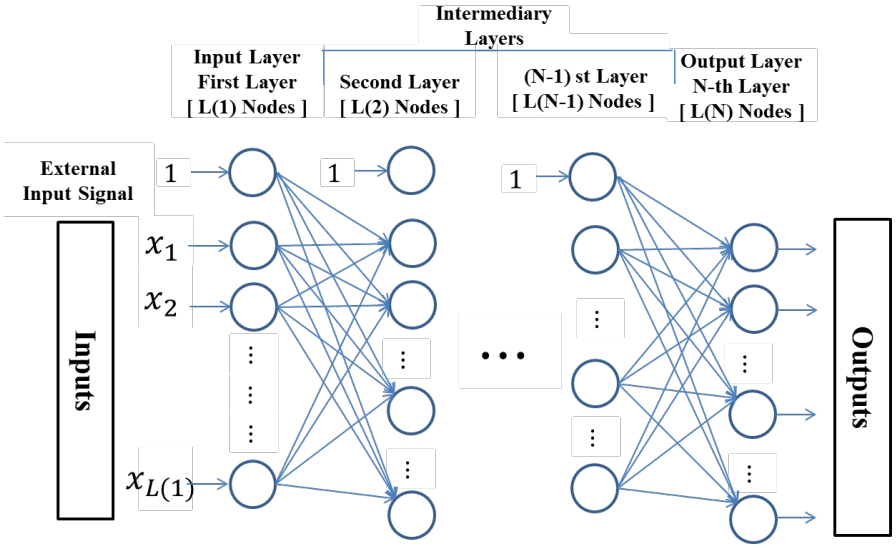


Figure 3.8.3.1 Multi-layered Neural Network

3.8.4 Example of Single Layer Neural Network

Suppose that a corporation operates multiple golf courses over Japan. The company is planning to build a new golf course and is interested in identifying conditions so as to make a new golf course profitable. The following three conditions are considered:

x_1: $high\ class(=1)\ or\ not\ (=0)$

x_2: $between\ 5km\ distance\ to\ a\ throughway\ entrance/exit\ (=1) or\ not\ (=0)$

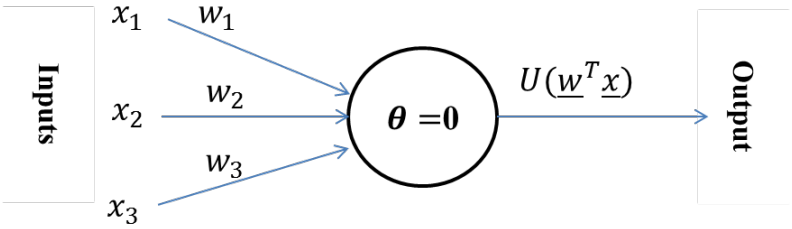
x_3: $hot\ spring\ available\ in\ the\ club\ house(=1)\ or\ not\ (=0)$

A survey among customers is conducted, yielding the next table, where

$y = 1$ means that the respondent will frequent the new golf course and $y = 0$, otherwise.

k	x_1	x_2	x_3	y
1	1	0	0	0
2	1	0	1	1
3	1	1	0	1
4	1	1	1	1

Of interest is to find the weight w_i $(i = 1, 2, 3)$ for the following single layer neural network so that the resulting neural network would reflect he table well.



A basic learning rule for this single layer neural network is to repeat the following process for $k = 1, 2, 3, 4$ starting with $\underline{w}^T{}_{old} = (0, 0, 0)$.

$$\underline{w}^T{}_{new} = \underline{w}^T{}_{old} - \eta\left\{U\left(\underline{w}^T{}_{old}\underline{x}(k)\right) - y(k)\right\}\underline{x}^T(k) \text{until } \underline{w}^T{}_{new} = \underline{w}^T{}_{old} \quad (3.8.1)$$

Let $\eta = 1$. Each step for $k = 1 \; and \; 2$ is depicted below.

$k = 1$

$\underline{w}^T{}_{old} = (0, 0, 0), \quad \underline{x}^T(1) = (1, 0, 0) \Rightarrow \underline{w}^T{}_{old}\underline{x}(1) = 0$

$\Rightarrow U(0) - y(1) = 1 - 0 = 1 \Rightarrow \underline{w}^T{}_{new} = (0, 0, 0) - \underline{x}^T(1) = (-1, 0, 0)$, $\underline{w}^T{}_{old} \leftarrow \underline{w}^T{}_{new}$

$k = 2$

$\underline{w}^T{}_{old} = (-1, 0, 0), \; \underline{x}^T(2) = (1, 0, 1), \; \underline{w}^T{}_{old}\underline{x}(2) = -1$

$$\Rightarrow U(-1) - y(2) = 0 - 1 = -1 \Rightarrow \underline{w}^T{}_{new} = \underline{w}^T{}_{old} + \underline{x}^T(2) = (-1, 1, 1)$$

By repeating this process, the following table can be obtained with $\underline{x}^T = (-1, 1, 1)$.

# of Repeats	k	$\underline{w}^T_{old}$			$\underline{w}^T_{old}\underline{x}(k)$	$U\left(\underline{w}^T_{old}\underline{x}(k)\right)$	$U\left(\underline{w}^T_{old}\underline{x}(k)\right) - y(k)$	$\underline{w}^T_{new}$		
1	1	0	0	0	0	1	1	−1	0	0
	2	−1	0	0	−1	0	−1	0	0	1
	3	0	0	1	0	1	0	0	0	1
	4	0	0	1	1	1	1	0	0	1
2	1	0	0	1	0	1	1	−1	0	1
	2	−1	0	1	0	1	0	−1	0	1
	3	−1	0	1	−1	0	−1	0	1	1
	4	0	1	1	2	1	0	0	1	1
3	1	0	1	1	0	1	1	−1	1	1
	2	−1	1	1	0	1	0	−1	1	1
	3	−1	1	1	0	1	0	−1	1	1
	4	−1	1	1	1	1	0	−1	1	1
4	1	−1	1	1	−1	0	0	−1	1	1
	2	−1	1	1	0	1	0	−1	1	1
	3	−1	1	1	0	1	0	−1	1	1
	4	−1	1	1	1	1	0	−1	1	1

3.8.5 Neural Network Based on Sigmoid Function

A learning mechanism for neural networks should support the continuous learning in that minor changes in weights and thresholds result in only minor changes of the output. As the step function does not satisfy this condition, in order to achieve the continuous learning, the sigmoid function $\sigma(z) = (1 + \mathrm{e}^{-z})^{-1}$ has been introduced. The sigmoid function $\sigma(z)$ and the associated single-layered neural network is illustrated below.

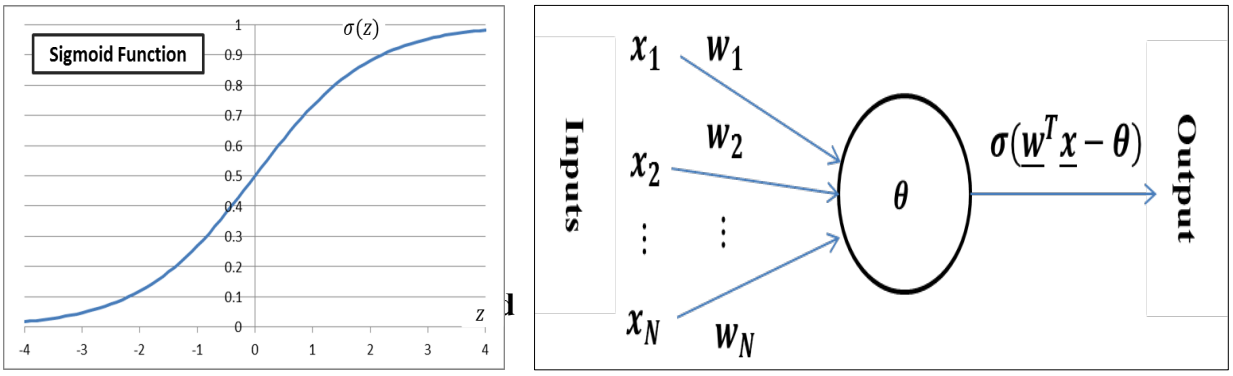


Figure 3.8.5.1 Sigmoid Function $\sigma(z)$ and Associated Single-layered Neural Network

When the sigmoid function is employed, one has to find a way to obtain the best weights. For this purpose, the steepest descent method is often employed. We sketch the method using a single layer neural network with the sigmoid as an example. The problem is then to minimize

$$E[w|x_0\ y_0] = \{y_0 - \sigma(wx_0)\}^2 = \left\{y_0 - \frac{1}{1+e^{-wx_0}}\right\}^2. \tag{3.8.2}$$

Let $x_0 = 1$ and $y_0 = \sigma(1) = 0.731059$ be a set of Learning Data. Starting with $\underline{w}^T{}_{old}$ which is randomly chosen, $E[w|x_0\ y_0]$ is approximated by a line. If the sign of the slope is positive, $\underline{w}^T{}_{old}$ should be decreased. If it is negative, $\underline{w}^T{}_{old}$ be increased, obtaining $\underline{w}^T{}_{new}$. Letting $\underline{w}^T{}_{old} \leftarrow \underline{w}^T{}_{new}$, this process should be repeated. More formally, with some $\eta > 0$, one has

$$\underline{w}^T{}_{new} = \underline{w}^T{}_{old} + \Delta w \ \text{ where } \ \Delta w = -\eta \frac{d}{dw} E(w|x_0\ y_0)|_{w=w_{old}}\,. \tag{3.8.3}$$

One step of this process is depicted below.

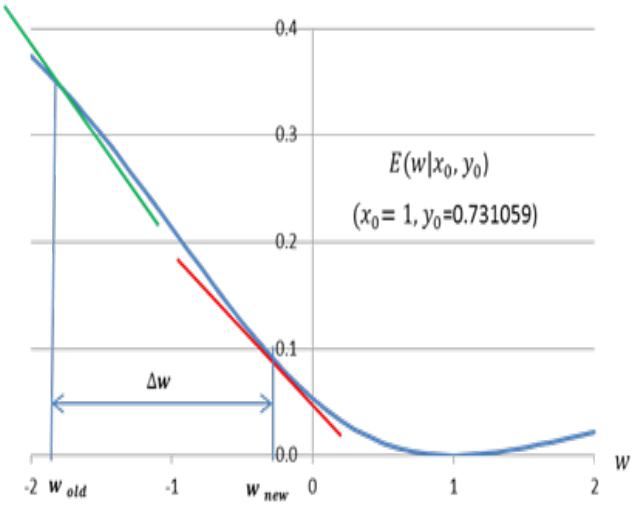


Figure 3.8.6.1 Steepest Descent Method

3.8.7 Multi-layered Neural Network with Sigmoid Function

We now consider a multi-layered neural network with the sigmoid function, as exhibited in Figure 3.8.7.1.

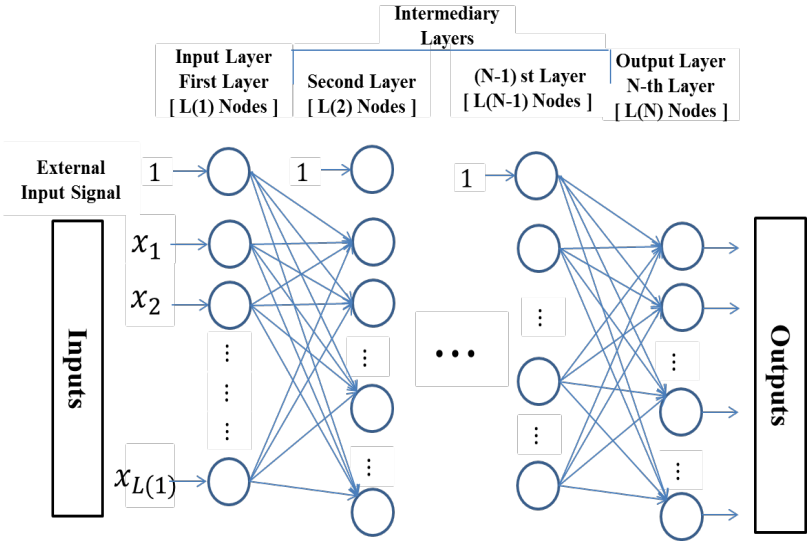


Figure 3.8.7.1 Multi-layered Neural Network with Sigmoid Function

Let x_i^k be the sum of all inputs at the i – th node in the k – th layer, which we write as *(k ,i)*. It should be noted that $x_i^k = x_i, i = 1, \cdots L(1)$. The output y_i^k is given as

$$y_i^k = \sigma(x_i^k) = (1 + e^{-x_i^k})^{-1} \quad . \tag{3.8.4}$$

Let $w_{i,j}^{k,k+1}$ be the weight associated with the link between (*k, i*) and (*k+1, j*). Then one has

$$x_j^{k+1} = \sum_{i=1}^{L(k)} w_{i,j}^{k,k+1} \times y_i^k \quad . \tag{3.8.5}$$

For notational convenience, we write, in a matrix form, $\underline{\underline{w}}\,(k, k+1) = \left[w_{i,j}^{k,k+1} \right]$ and

$\underline{\underline{w}} = \left\{\underline{\underline{w}}(1,2), \cdots, \underline{\underline{w}}(N-1,N)\right\}$. Given the input vector $\underline{x} = \left[x_1, \cdots, x_{L(1)}\right]^T$, $y^N = \left[y_1, \cdots, y_{L(N)}\right]^T$ and the output vector $y^N = \left[y_1, \cdots, y_{L(N)}\right]^T$ to be learned, the problem is to find $\underline{\underline{w}}^*$, which minimizes

$$\frac{1}{2} E\left(\underline{\underline{w}} | x, y\right) = \frac{1}{2} \sum_{j=1}^{L(N)} (\delta_j^N)^2 \,;\; \delta_j^N = y_j^N - y_j \qquad . \qquad (3.8.6)$$

In order to solve this problem, the steepest descent $E\left(\underline{\underline{w}} | \underline{x}, y\right)$ at $\underline{\underline{w}}$ is approximated by a hyperplane. The algorithm now can be summarized as follows.

[0] Choose the elements of $\underline{\underline{w}}$ at a random from $U[0,1]$. Set $\underline{\underline{w}}^T{}_{old} \leftarrow \underline{\underline{w}}$.

[1] LOOP: Starting with $\underline{x} = \left[x_1, \cdots, x_{L(1)}\right]^T$, compute y_i^k for all $1 \le k \le N$ and $1 \le k \le L(k)$ based on Equations (1) and (2).

[2] Obtain δ_j^k based on

$$\delta_j^k = \begin{cases} y_j^N - y_j & if\ k = N \\ \sum_{r=1}^{L(k+1)} w_{j,r}^{k,k+1}\ \delta_r^{k+1} & otherwise \end{cases} \qquad (3.8.7)$$

[3] Starting with $k = N$, repeat the next procedure until $k = 2$, yielding $\underline{\underline{w}}(new)$.

$$w_{i,j}^{k-1,k}(new) = w_{i,j}^{k-1,k}(old) + \Delta\, w_{i,j}^{k-1,k} \;;$$

$$\Delta w_{i,j}^{k-1,k} = -\eta \times \delta_j^k y_j^k (1 - y_j^k) y_i^{k-1} \tag{3.8.8}$$

[4] Using $\underline{w}(new)$, compute $y_j^k 1 \leq k \leq N$ and $1 \leq k \leq L(k)$ based on Equation (3.8.4).

[5] Evaluate $E(\underline{w}(new)|x, y)/2$. If it is less than sufficiently small $\varepsilon > 0$, set $\underline{\underline{w^*}} \leftarrow \underline{\underline{w}}(old)$ and continue. Otherwise set $\underline{\underline{w}}(old) \leftarrow \underline{\underline{w}}(new)$ and go back to LOOP.

Chapter 4

Development of Segmentation Algorithms for Identifying Smartphone Applications with Sustainable Popularity Ranking: Approach I Based on Six Prevalent Segmentation Algorithms

We are now in a position to propose two segmentation algorithms for identifying smartphone applications with sustainable popularity ranking based on the results of K different segmentation algorithms. For segmentation algorithm k, the corresponding indices and the flag are denoted by $Recall(k)$, $Precision(k)$, $f(k)$ and $I_{Seg(k)}(a)$, $k = 1, \cdots, K$. Similarly, for two combined segmentation algorithms, we write $Com\text{-}Recall(q)$, $Com\text{-}Precision(q)$, $Com\text{-}f(q)$ and $I_{Com-Seg(q)}(a)$, $q = \mathrm{I}, \mathrm{II}$.

4.1 Combined Segmentation Algorithm I : $(\mathcal{V}^*, \mathcal{W}^*)$ Voting Approach

[1] Construct K different segmentation algorithms based on $\mathrm{LD}(\tau)$

[2] Apply the constructed segmentation algorithms to $\mathrm{TD}(\tau)$, yielding $f(k)$, $k = 1, \cdots, K$.

[3] Choose V segmentation algorithms in the descending order of $f(k)$. Let $\{k_1, \cdots, k_V\}$ be the set of indices for the chosen segmentation algorithms.

[4] Set $I_{Com-Seg(\mathrm{I})}(a) = 1$ if $\sum_{j=1}^{V} I_{Seg(k_j)}(a) \geq \mathrm{W}$; and $I_{Com-Seg(\mathrm{I})}(a) = 0$, otherwise.

[5] Apply [4] to $\mathrm{TD}_1(\tau)$ for obtaining $Com - Recall(I|V, W)$ and $Com\text{-}Precision(I|V, W)$.

[6] Find

$(V^*, W^*) = \mathrm{argmax}\{Com\text{-}Precision(I|V, W)$ subject to $Com\text{-}Recall(I|V, W) \geq 0.6\}$

In this algorithm, K different segmentation algorithms are first ordered in the descending order of $f(k)$ and choose the first V of them in this order. The combined flag $I_{Com-Seg(\mathrm{I})}(a)$ is set to be 1 if the sum of $I_{Seg(k_j)}(a)$ over $j = 1, \cdots, V$, from which $Com - Recall(\mathrm{I}|\mathrm{V}, \mathrm{W})$ and $Com\text{-}Precision(\mathrm{I}|\mathrm{V}, \mathrm{W})$ can be computed. The underlying two parameters V and W are then optimized by maximizing $Com\text{-}Precision(\mathrm{I}|\mathrm{V}, \mathrm{W})$ subject to $Com\text{-}Recall(\mathrm{I}|\mathrm{V}, \mathrm{W}) \geq 0.6$.

4.2 Combined Segmentation Algorithm II : $(\underline{\boldsymbol{\alpha}}^*, \boldsymbol{\beta}^*)$ Optimal Linear Combination

[1] Construct K different segmentation algorithms based on $\mathrm{LD}(\tau)$.

[2] Apply the constructed segmentation algorithms to $\mathrm{LD}_1(\tau)$, yielding $I_{Seg(k)}(a),\ k = 1, \cdots, K$.

[3] For $\underline{\alpha}^T = [\alpha_1, \cdots, \alpha_K] \geq 0$ with $\sum_{i=1}^{K} \alpha_i = 1$, let

$$I_{Com-Seg(\mathrm{II}|\,\underline{\alpha},\beta)}(a) = \begin{cases} 1 & \text{if } \sum_{k=1}^{K} \alpha_i \times I_{Seg(k)}(a) \geq \beta \\ 0 & otherwise \end{cases} . \qquad (4.2.1)$$

[4] Given $\underline{\alpha}^T$, compute $Com\text{-}Recall(\mathrm{II}|\,\underline{\alpha}, \beta)$ based on (3.1.1) and (4.2.1).

[5] Let $FR(\gamma|\beta) = \{\underline{\alpha}: Com\text{-}Recall(\mathrm{II}|\,\underline{\alpha}, \beta) \geq \beta,\ \underline{\alpha} \geq 0,\ \sum_{i=1}^{K} \alpha_i = 1\}$.

[6] If $FR(\gamma|\beta) = \phi, then$ solve the following optimization problem.

$\underline{\alpha}^* = \arg\max\{Com\text{-}Recall(\text{II}|\underline{\alpha}, \beta)$ subject to $\underline{\alpha} \geq 0$, and $\sum_{i=1}^{K} \alpha_i = 1\}$

[7] Otherwise, solve the following optimization problem.

$\underline{\alpha}^* = \arg\max\{Com\text{-}Precision(\text{II}|\underline{\alpha}, \beta)$ subject to $\underline{\alpha} \in FR(\gamma|\beta)\}$

[8] Set $I_{Com-Seg(\text{II})}(a) = I_{Com-Seg(\text{II},|\underline{\alpha}^*,\beta)}(a)$.

[9] Apply [8] to $\text{TD}_1(\tau)$ for obtaining $Com\text{ - }Recall(\text{II}|\underline{\alpha}^*, \beta)$ and $Com\text{-}Precision(\text{II}|\underline{\alpha}^*, \beta)$.

[10] Find

$\beta^* = \arg\max_{\beta}\{Com - Precision(\text{II}|\underline{\alpha}^*, \beta)$ subject to $Com - Recall(\text{II}|\underline{\alpha}^*, \beta) \geq 0.6\}$.

Instead of choosing a promising subset of K different segmentation algorithms, this algorithm utilizes all of them by taking a linear combination $\sum_{k=1}^{K} \alpha_i \times I_{Seg(k)}(a)$ and then setting the combines segmentation function $I_{Com-Seg(\text{II},\ \underline{\alpha})}(a)$ to be 1 if the combined value is greater than or equal to β, and 0 otherwise. For any $\underline{\alpha}^T = [\alpha_1, \cdots, \alpha_K] \geq 0$ with $\sum_{i=1}^{K} \alpha_i = 1$, if $Com\text{-}Recall(\text{II}, \underline{\alpha})$ is below β, choose $\underline{\alpha}^*$ by maximizing $Com - Recall(\text{II}, \underline{\alpha})$. Otherwise, choose $\underline{\alpha}^*$ by maximizing $Com - Precision(\text{II}, \underline{\alpha})$ subject to $Com\text{-}Recall(\text{II}, \underline{\alpha}) \geq \beta$. The second parameter β is then optimized by maximizing $Com - Precision(\text{II}|\underline{\alpha}^*, \beta)$ subject to $Com - Recall(\text{II}|\underline{\alpha}^*, \beta) \geq 0.6$.

4.3 Numerical Results for Japanese Free Games

For K different segmentation algorithms needed to develop the two combined algorithms discussed in Sections 4.1 and 4.2, we employ (LR), Decision Tree

(DT), Support Vector Machine (SVM), Random Forest (RF), Neural Network (NNet) and Naïve Bayes Classifier (NBC), described in Chapter 3, provided by the R language.

In Tables 4.3.1 (a) through (c), the three data sets $\mathrm{LD}(\tau)$, $\mathrm{TD}(\tau) = \mathrm{LD}_1(\tau)$ and $\mathrm{TD}_1(\tau)$ are summarized respectively over 12 periods. One sees in Table 4.3.1 (a) that about 1500 free game applications constitute $\mathrm{LD}(\tau)$ for each period with the hitting rate of a random choice around 0.24. These figures are about 128 and 0.23 for $\mathrm{TD}(\tau) = \mathrm{LD}_1(\tau)$, and 124 and 0.22 for $\mathrm{TD}_1(\tau)$ as shown in Tables 4.3.1 (b) and (c) respectively. We note that the expected accuracy of randomly guessing a smartphone application to be with sustainable popularity ranking would be around 0.22～0.24, and the two segmentation algorithms proposed in this thesis should achieve its accuracy much beyond this level.

Table 4.3.1 (a) Learning Data: $\mathrm{LD}(\tau)$

Period(τ)	Starting Month (t_0)	(a) $\lvert\mathrm{LD}(\tau)\rvert$	(b) $\sum_{a\in \mathrm{LD}(\tau)} I_{True}(a)$	$(b)/(a)$
1	2013/8	1576	392	0.249
2	2013/9	1572	391	0.249
3	2013/10	1567	396	0.253
4	2013/11	1576	393	0.249
5	2013/12	1579	390	0.247
6	2014/1	1589	379	0.239
7	2014/2	1591	373	0.234
8	2014/3	1591	366	0.230
9	2014/4	1581	371	0.235
10	2014/5	1568	359	0.229
11	2014/6	1552	348	0.224
12	2014/7	1533	342	0.223
Average		1572.92	375.00	0.238

Table 4.3.1 (b) Testing/Learning Data: $\mathbf{TD(\tau) = LD_1(\tau)}$

Period(τ)	Starting Month (t_0)	(a) $\lvert LD_1(\tau)\rvert$	(b) $\sum_{a \in LD_1(\tau)} I_{True}(a)$	$^{(b)}/_{(a)}$
1	2013/8	130	33	0.254
2	2013/9	120	31	0.258
3	2013/10	135	26	0.193
4	2013/11	142	34	0.239
5	2013/12	138	28	0.203
6	2014/1	105	19	0.181
7	2014/2	123	26	0.211
8	2014/3	131	38	0.290
9	2014/4	126	31	0.246
10	2014/5	132	30	0.227
11	2014/6	119	23	0.193
12	2014/7	131	28	0.214
Average		127.67	28.92	0.226

Table 4.3.1 (c) Testing Data: $\mathbf{TD_1(\tau)}$

Period(τ)	Starting Month (t_0)	(a) $\lvert TD_1(\tau)\rvert$	(b) $\sum_{a \in TD_1(\tau)} I_{True}(a)$	$^{(b)}/_{(a)}$
1	2013/8	120	31	0.258
2	2013/9	135	26	0.193
3	2013/10	142	34	0.239
4	2013/11	138	28	0.203
5	2013/12	105	19	0.181
6	2014/1	123	26	0.211
7	2014/2	131	38	0.290
8	2014/3	126	31	0.246
9	2014/4	132	30	0.227
10	2014/5	119	23	0.193
11	2014/6	131	28	0.214
12	2014/7	80	15	0.188
Average		123.50	27.42	0.220

For each segmentation algorithm, the Learning Data $LD(\tau)$ is first used to determine the underlying parameter values. The established segmentation algorithm is

then applied to the Testing Data TD(τ). The performances of these segmentation algorithms are summarized in Table 4.3.2. One sees that all methods achieve, on the average, the accuracy of 0.7 or better. The balance between Recall and Precision is not well reflected by DT (0.454 vs.0.606) and SVM (0.800 vs.0.449), while LR (0.613 vs.0.565), RF (0.639 vs.0.562), NNet (0.599 vs.0.605) and NBC (0.583 vs. 0.658) satisfy the balance better. Considering the accuracy and the balance between Recall and Precision as well as the stability of the performance over the twelve periods, NNet seems to outperform all others.

Table 4.3.2 Performance of Individual Segmentation Algorithms

$(\boldsymbol{m_0 = 2, m_1 = 6\ ;\ l_1 - l_0 = 2})$

	LR			DT			RF			SVM			NNet			NBC		
Period(τ)	Rec	Pre	Acc	Rec	Pre	Acc	Rec	Pre	Acc	Rec	Pre	Acc	Rec	Pre	Acc	Rec	Pre	Acc
1	0.581	0.818	0.858	0.355	0.917	0.825	0.613	0.792	0.858	0.581	0.818	0.858	0.516	0.889	0.858	0.548	0.850	0.858
2	0.462	0.667	0.852	0.423	0.611	0.837	0.654	0.531	0.822	0.846	0.489	0.800	0.500	0.765	0.874	0.500	0.684	0.859
3	0.647	0.579	0.803	0.412	0.483	0.754	0.647	0.550	0.789	0.941	0.464	0.725	0.676	0.561	0.796	0.647	0.629	0.824
4	0.679	0.514	0.804	0.571	0.500	0.797	0.821	0.434	0.746	0.929	0.286	0.514	0.821	0.511	0.804	0.786	0.440	0.754
5	0.789	0.455	0.790	0.579	0.500	0.819	0.684	0.722	0.895	0.842	0.267	0.552	0.684	0.464	0.800	0.789	0.536	0.838
6	0.769	0.556	0.821	0.538	0.452	0.764	0.769	0.392	0.699	0.846	0.333	0.610	0.769	0.392	0.699	0.731	0.576	0.829
7	0.579	0.647	0.786	0.316	0.800	0.779	0.421	0.696	0.779	0.684	0.542	0.740	0.553	0.700	0.802	0.447	0.654	0.771
8	0.452	0.560	0.778	0.323	0.833	0.817	0.452	0.583	0.786	0.645	0.488	0.746	0.452	0.737	0.825	0.387	0.857	0.833
9	0.533	0.457	0.750	0.500	0.600	0.811	0.667	0.476	0.758	0.767	0.390	0.674	0.533	0.533	0.788	0.533	0.593	0.811
10	0.522	0.480	0.798	0.304	0.467	0.798	0.565	0.406	0.756	0.870	0.385	0.706	0.478	0.440	0.782	0.565	0.619	0.849
11	0.607	0.607	0.832	0.393	0.688	0.832	0.571	0.667	0.847	0.714	0.541	0.809	0.536	0.682	0.847	0.393	0.688	0.832
12	0.733	0.440	0.775	0.733	0.423	0.762	0.800	0.500	0.812	0.933	0.389	0.712	0.667	0.588	0.850	0.667	0.769	0.900
Average	0.613	0.565	0.804	0.454	0.606	0.800	0.639	0.562	0.796	0.800	0.449	0.704	0.599	0.605	0.810	0.583	0.658	0.830
Standard Deviation	0.108	0.105	0.031	0.126	0.160	0.028	0.120	0.126	0.052	0.115	0.142	0.099	0.115	0.144	0.044	0.136	0.118	0.037

Table 4.3.3 exhibits the performance results of Combined Segmentation Algorithm I: (V^*, W^*) Voting Approach, where the values of (V^*, W^*) and the selected method(s) are also provided for each period. The algorithm achieves, on the average, Recall of 0.623 and Precision of 0.621, superseding NNet, which is the best single method out of the six methods, with 0.599 and 0.605 respectively, while Accuracy of 0.821 which is better than 0.810 by NNet. In Table 4.3.4, the performance results of

Combined Segmentation Algorithm II: $(\underline{\alpha}^*, \beta)$ Optimal Linear Combination Approach are summarized, where the values of β^* and the optimal weights associated with the six methods are also provided for each period. One sees that they are more or less comparable with those of Combined Segmentation Algorithm I: (V^*, W^*) Voting Approach in Table 4.3.3.

Table 4.3.3 Performance of Combined Segmentation Algorithm I: $(\mathbf{V}^*, \mathbf{W}^*)$ Voting Approach

Period(τ)	V*	W*	Selected Method						Recall	Precision	Accuracy
			LR	DT	RF	SVM	NNet	NBC			
1	2	1					○	○	0.613	0.864	0.875
2	4	2			○	○	○	○	0.654	0.586	0.844
3	2	1					○	○	0.765	0.578	0.810
4	2	2	○					○	0.643	0.545	0.819
5	2	2			○		○		0.632	0.706	0.886
6	2	2	○					○	0.654	0.708	0.870
7	2	2			○			○	0.421	0.696	0.779
8	6	3	○	○	○	○	○	○	0.516	0.667	0.817
9	1	1				○			0.767	0.390	0.674
10	4	3	○		○		○	○	0.478	0.500	0.807
11	3	2				○	○	○	0.536	0.714	0.855
12	1	1			○				0.800	0.500	0.813
Average									0.623	0.621	0.821
Standard Deviation									0.119	0.128	0.056

Table 4.3.4 Performance of Combined Segmentation Algorithm II: $(\boldsymbol{\alpha}^*, \boldsymbol{\beta})$ Optimal Linear Combination Approach

Period(τ)	β^*	LR	DT	RF	SVM	NNet	NBC	Recall	Precision	Accuracy
1	0.5	0.0	0.0	0.1	0.0	0.5	0.4	0.548	0.850	0.858
2	0.5	0.1	0.4	0.1	0.0	0.4	0.0	0.500	0.684	0.859
3	0.5	0.0	0.0	0.1	0.0	0.5	0.4	0.735	0.568	0.803
4	0.6	0.1	0.4	0.0	0.0	0.0	0.5	0.714	0.526	0.812
5	0.7	0.1	0.3	0.4	0.0	0.2	0.0	0.632	0.750	0.895
6	0.7	0.1	0.3	0.0	0.0	0.1	0.5	0.654	0.680	0.862
7	0.7	0.0	0.2	0.2	0.0	0.2	0.4	0.421	0.727	0.786
8	0.5	0.1	0.2	0.0	0.3	0.2	0.2	0.484	0.750	0.833
9	0.5	0.5	0.0	0.1	0.4	0.0	0.0	0.700	0.457	0.742
10	0.7	0.2	0.1	0.1	0.1	0.3	0.2	0.391	0.450	0.790
11	0.5	0.1	0.0	0.0	0.0	0.4	0.5	0.500	0.667	0.840
12	0.5	0.0	0.0	0.3	0.2	0.2	0.3	0.800	0.500	0.813
Average								0.590	0.634	0.824
Standard Deviation								0.133	0.130	0.042

In order to reveal some subtle characteristics of smartphone application with sustainable popularity ranking, it is necessary to look into the optimally determined parameter values of the individual segmentation algorithms. In Table 4.3.5, for example, the regression coefficients and the intercept for LR are shown over the twelve periods.

One realizes that Act-Count-m-10 and Act-Count-m-30 are always positive. Act-Count-Change-f-late20 and Ins-Count-f-30 are also almost always positive except two Period. This means that if a smartphone application has a large number of activations in three months after the introduction into the market by those users who are male in 10's and 30's, it is likely to have the property of sustainable popularity ranking. The same conclusion can be made in a slightly weak manner, when it has rapid increase of the number of activations in three months after the introduction into the market by those users who are female and in late 20's, or it is installed by many users who are female and in 30's in three months after the introduction into the market.

Focusing on the trend observed over the last four periods, it can be seen that $Act-Count-Change-m-10$, $Act-Count-Change-m-late20$ and $Act-Count-Change-m-early20$ have been negative, while $Possess-Count-Change-m-40$ and $Possess-Count-Change-f-30$ have been positive. This implies that, if a smartphone application has the rapid increase of the number of activations in three months after the introduction into the market by those users who are male and in 10's or 20's, it is less likely that the application has the property of sustainable popularity ranking. On the contrary, the recent trend also states that if a smartphone application experiences the rapid increase of the number of possessions by those users who are male and in 40's or female in 30's in three months after the introduction into the market, it is more likely that the application has the property of sustainable popularity ranking.

In order to identify the impact of individual components of the application profile vector on sustainable popularity ranking more deeply, we develop five different algorithms based only on LR and DT in the next chapter.

Table 4.3.5 Estimated Coefficients in Logit Method

	Positive	Negative	Not Selected	Period (1)	Period (2)	Period (3)	Period (4)	Period (5)	Period (6)	Period (7)	Period (8)	Period (9)	Period (10)	Period (11)	Period (12)
(Intercept)	0	12	0	-2.94	-2.96	-2.97	-2.91	-2.93	-2.96	-3.03	-2.98	-2.86	-2.80	-2.80	-2.80
Act-Count-Change-m-10	0	11	1	-1.38		-1.26	-1.30	-2.04	-2.18	-2.89	-2.71	-1.89	-1.77	-1.56	-1.33
Act-Count-Change-m-late20	0	7	5		-1.52				-1.24	-1.47		-1.75	-2.35	-1.51	-2.31
Act-Count-Change-m-early20	0	6	6						-1.42		-2.45	-2.89	-1.16	-2.84	-2.39
Possess-Count-Change-m-40	7	0	5						1.70	2.02	1.95	2.56	2.68	2.32	1.88
Possess-Count-Change-f-30	8	0	4					2.28	2.06	1.89	1.98	3.00	2.81	2.62	2.69
Ins-Count-m-early20	9	0	3	2.53	1.62	1.52	1.86	2.66	3.10	2.51	1.56	1.71			
Act-Count-Change-f-late20	10	0	2	1.63	2.08	1.76	1.45		1.97	2.35	1.43	1.51	1.49	1.54	
Ins-Count-f-30	10	0	2	3.21	2.05	1.71	1.90	1.83	2.28	1.47	1.95	2.45	2.22		
Act-Count-m-10	12	0	0	2.03	1.95	2.42	2.43	3.21	3.07	2.94	2.22	1.90	1.88	1.94	2.36
Act-Count-m-30	12	0	0	1.02	1.35	1.50	1.50	1.62	1.43	1.55	1.64	1.60	1.62	1.75	0.87

Chapter 5

Development of Segmentation Algorithms for Identifying Smartphone Applications with Sustainable Popularity Ranking: Approach II Based on LR and DT

In this chapter, the two results of LR and DT are combined in five different ways, resulting in five segmentation algorithms for identifying smartphone applications with sustainable popularity ranking.

For notational convenience, we introduce a generic variable $SEG \in \{LR, DT\}$ where LR and DT stand for Logistic Regression and Decision Tree, respectively. For $a \in A(T)$, let $P_{Seg}(a)$ be the estimated probability that application a satisfies the sustainable popularity condition. Given $z \in (0,1)$, the resulting indicator function $I_{Seg}(a,z)$ is then defined as

$$I_{Seg}(a|z) = \begin{cases} 1 & if\ P_{Seg}(a) \geq z \\ 0 & else \end{cases} \quad . \tag{5.0.1}$$

Let $Recall_{Seg}(z)$ and $Precision_{Seg}(z)$ be the recall and the precision associated with $I_{Seg}(a|z)$, as defined in (5.0.1). For $SEG \in \{LR, DT\}$, then the optimal segmentation level z^* is determined by solving

$$z^* = \arg\max_{0<z<1}\{Precision_{Seg}\ (z) \text{ subject to } Recall_{Seg}\ (z)\ \geq 0.6\}\quad . \qquad (5.0.2)$$

We note that z^* maximizes $Precision_{Seg}\ (z)$ while satisfying $Recall_{Seg}(z)\ \geq 0.6$.

In what follows, we designate LR as Algorithm I and DT as Algorithm II. Accordingly, five new algorithms constructed from LR and DT are called Algorithm III through Algorithm VII.

5.1 Combined Segmentation Algorithm III: Determination of Optimal Threshold Based on Average

Algorithm III : $I_{III}\ (a|\ z^*) = I_{Ave}\ (a|z^*)$

[1] $P_{Ave}\ (a) = \{P_{LR}\ (a) + P_{DT}(a)\}/2$

[2] $I_{Ave}(a|z) = \begin{cases} 1 & \text{if } P_{Seg}(a) \geq z \\ 0 & else \end{cases}$

[3] $z^* = \arg\max_{0<z<1}\{Precision_{Ave}\ (z) \text{ subject to } Recall_{Ave}\ (z)\ \geq 0.6\}$

Algorithm III first takes the average of the two probabilities $P_{LR}\ (a)$ and $P_{DT}(a)$, yielding $P_{Ave}\ (a)$. Given $z \in (0,1)$, this enables one to define the flag function $I_{Ave}(a|z)$ as in [2]. The threshold z is then optimized by maximizing $Precision_{Ave}\ (z)$ subject to $Recall_{Ave}\ (z)\ \geq 0.6$ over $z \in (0,1)$.

5.2 Combined Segmentation Algorithm IV: Determination of Optimal Threshold Based on Maximum

Algorithm IV : $\mathrm{I_{IV}}\ (\mathrm{a}|\ \mathrm{z}^*) = \mathrm{I_{Max}}\ (\mathrm{a}|\mathrm{z}^*)$

[1] $\mathrm{P_{Max}(a) = \max\{P_{LR}\ (a), P_{DT}(a)\}}$

[2] $\mathrm{I_{Max}(a|z)} = \begin{cases} 1 & \text{if } P_{Max}(a) \geq z \\ 0 & else \end{cases}$

[3] $\mathrm{z^* = \arg\max_{0<z<1}\{Precision_{Max}\ (z)\ subject\ to\ Recall_{Max}\ (z)\ \geq 0.6\}}$

In Algorithm IV, the maximum of the two probabilities $P_{LR}\,(a)\ and\ P_{DT}(a)$ is used instead of the average for yielding $P_{Max}(a)\ in$ [1] $and\ I_{Max}(a|z)$ in [2]. The optimal z^* is determined as in Algorithm III.

5.3 Combined Segmentation Algorithm V: Determination of Optimal Threshold Based on Minimum

Algorithm V : $I_V\,(a|z^*) = I_{Min}\,(a|z^*)$

[1] $P_{Min}(a) = min\{P_{LR}\,(a), P_{DT}(a)\}$

[2] $I_{Min}(a|z) = \begin{cases} 1 & \text{if } P_{Min}(a) \geq z \\ 0 & else \end{cases}$

[3] $z^* = \arg\max_{0<z<1}\{Precision_{Min}\ (z)\ subject\ to\ Recall_{Min}\ (z)\ \geq 0.6\}$

Algorithm V is similar to Algorithm IV, where the maximum of the two probabilities $P_{LR}\,(a)\ and\ P_{DT}(a)$ is replaced by the minimum of them.

5.4 Combined Segmentation Algorithm VI: Segmentation Based on Logical Operation ∧

Algorithm VI: $I_{VI}\,(a|z^*) = I_{And}\,(a|z^*)$

[1] $I_{And}(a|z) = min\{\,I_{LR}\,(a|z)\,,\ I_{DT}(a|z)\,\}$

[2] $z^* = \arg\max_{0<z<1}\{Precision_{And}\ (z)\ subject\ to\ Recall_{And}\ (z)\ \geq 0.6\}$

Unlike the previous three algorithms, Algorithm VI takes the segmentation functions $I_{LR}(a|z)\ and\ I_{DT}(a|z)$ first. The combined segmentation function $I_{And}(a|z)$ is then constructed by computing the logical operation $\wedge$ between the two.

5.5 Combined Segmentation Algorithm VII: Segmentation Based on Logical Operation $\vee$

Algorithm VII : $I_{VII}(a|\ z^*) = I_{Or}(a|z^*)$

[1] $I_{Or}(a|z) = max\{\ I_{LR}(a|z)\ ,\ I_{DT}(a|z)\ \}$

[2] $z^* = \arg\max_{0<z<1}\{Precision_{Or}(z)\ subject\ to\ Recall_{Or}(z)\ \geq 0.6\}$

Algorithm VII is similar to Algorithm VI, where the logical operation $\wedge$ is replaced by $\vee$.

5.6 Numerical Results for Japanese Free Games

In this section, we first present numerical results obtained by applying the seven algorithms developed in the previous sections 5.1 through 5.5 to a set of smartphone applications in the category of Japanese free games denoted by FGC-ALL. The set of explanatory variables used in LR and DT are listed in Table 5.6.1. In order to balance the role of each of the explanatory variables in the two algorithms, they are standardized in the following manner. Let $x(a)$ be an explanatory variable associated with application $a \in$ FGC-ALL and define

$$\mu_{FGC-ALL}(x) = \frac{1}{|\text{FGC-ALL}|}\sum_{a\in\text{FGC-ALL}} x(a) \tag{5.6.1}$$

and

$$\sigma_{FGC-ALL}(x) = \sqrt{\frac{1}{|\text{FGC-ALL}|}\sum_{a\in \text{FGC-ALL}}\{x(a) - \mu_{FGC-ALL}(x)\}^2} \quad . \qquad (5.6.2)$$

The value $x(a)$ is then standardized by

$$\hat{x}(a) = \frac{x(a)-\mu_{FGC-ALL}(x)}{\sigma_{FGC-ALL}(x)} \quad . \qquad (5.6.3)$$

Table 5.6.1 List of Explanatory Variable

ID	Explanatory Variable	ID	Explanatory Variable	ID	Explanatory Variable	ID	Explanatory Variable
X1	Act_Count_f_10	X25	Unins_Count_f_10	X49	Act_Count_Change_f_10	X73	Unins_Count_Change_f_10
X2	Act_Count_f_early20	X26	Unins_Count_f_early20	X50	Act_Count_Change_f_early20	X74	Unins_Count_Change_f_early20
X3	Act_Count_f_late20	X27	Unins_Count_f_late20	X51	Act_Count_Change_f_late20	X75	Unins_Count_Change_f_late20
X4	Act_Count_f_30	X28	Unins_Count_f_30	X52	Act_Count_Change_f_30	X76	Unins_Count_Change_f_30
X5	Act_Count_f_40	X29	Unins_Count_f_40	X53	Act_Count_Change_f_40	X77	Unins_Count_Change_f_40
X6	Act_Count_f_over50	X30	Unins_Count_f_over50	X54	Act_Count_Change_f_over50	X78	Unins_Count_Change_f_over50
X7	Act_Count_m_10	X31	Unins_Count_m_10	X55	Act_Count_Change_m_10	X79	Unins_Count_Change_m_10
X8	Act_Count_m_early20	X32	Unins_Count_m_early20	X56	Act_Count_Change_m_early20	X80	Unins_Count_Change_m_early20
X9	Act_Count_m_late20	X33	Unins_Count_m_late20	X57	Act_Count_Change_m_late20	X81	Unins_Count_Change_m_late20
X10	Act_Count_m_30	X34	Unins_Count_m_30	X58	Act_Count_Change_m_30	X82	Unins_Count_Change_m_30
X11	Act_Count_m_40	X35	Unins_Count_m_40	X59	Act_Count_Change_m_40	X83	Unins_Count_Change_m_40
X12	Act_Count_m_over50	X36	Unins_Count_m_over50	X60	Act_Count_Change_m_over50	X84	Unins_Count_Change_m_over50
X13	Ins_Count_f_10	X37	Possess_Count_f_10	X61	Ins_Count_Change_f_10	X85	Possess_Count_Change_f_10
X14	Ins_Count_f_early20	X38	Possess_Count_f_early20	X62	Ins_Count_Change_f_early20	X86	Possess_Count_Change_f_early20
X15	Ins_Count_f_late20	X39	Possess_Count_f_late20	X63	Ins_Count_Change_f_late20	X87	Possess_Count_Change_f_late20
X16	Ins_Count_f_30	X40	Possess_Count_f_30	X64	Ins_Count_Change_f_30	X88	Possess_Count_Change_f_30
X17	Ins_Count_f_40	X41	Possess_Count_f_40	X65	Ins_Count_Change_f_40	X89	Possess_Count_Change_f_40
X18	Ins_Count_f_over50	X42	Possess_Count_f_over50	X66	Ins_Count_Change_f_over50	X90	Possess_Count_Change_f_over50
X19	Ins_Count_m_10	X43	Possess_Count_m_10	X67	Ins_Count_Change_m_10	X91	Possess_Count_Change_m_10
X20	Ins_Count_m_early20	X44	Possess_Count_m_early20	X68	Ins_Count_Change_m_early20	X92	Possess_Count_Change_m_early20
X21	Ins_Count_m_late20	X45	Possess_Count_m_late20	X69	Ins_Count_Change_m_late20	X93	Possess_Count_Change_m_late20
X22	Ins_Count_m_30	X46	Possess_Count_m_30	X70	Ins_Count_Change_m_30	X94	Possess_Count_Change_m_30
X23	Ins_Count_m_40	X47	Possess_Count_m_40	X71	Ins_Count_Change_m_40	X95	Possess_Count_Change_m_40
X24	Ins_Count_m_over50	X48	Possess_Count_m_over50	X72	Ins_Count_Change_m_over50	X96	Possess_Count_Change_m_over50

Table 5.6.2 below exhibits the optimal segmentation level z* obtained by (5.0.2) for each of the seven algorithms. One sees that Algorithm IV imposes the severest segmentation level of 0.600, while Algorithm V presents the weakest segmentation criterion of 0.210.

Table 5.6.2 The Optimal Segmentation Level z* for FGC-ALL

Algorithms		FGC-ALL
		z^*
Algorithm I	*Logistic Regression*	0.530
Algorithm II	*Decision Tree*	0.310
Algorithm III	*Average Model*	0.430
Algorithm IV	*Max Model*	0.600
Algorithm V	*Min Model*	0.210
Algorithm VI	*And Model*	0.530
Algorithm VII	*Or Model*	0.310

Using z* in Table 5.6.2, the segmentation results are summarized in Table 5.6.3, where the value 1 (the value 0) under I_{end}-FGC-ALL in the third column (the fourth column) implies the number of applications in FGC-ALL which satisfy (do not satisfy) the end condition of (2.3.12). These results are decomposed into subcategories of Japanese free games in the fifth through twelfth columns, where I_{end}-FGC-RP, I_{end}-FGC-CAS, I_{end}-FGC-PZ and I_{end}-FGC-ACT are defined for subcategories Role Play, Casual, Puzzle and Action. The resulting Precision and Recall are summarized in Table 5.6.4.

It can be seen that Algorithm III works best for FGC-ALL, achieving 0.647 for Precision and 0.602 for Recall, followed by Algorithm IV with Precision of 0.637 and Recall of 0.608. It may be worth noting that Algorithm VI works poorly for the Testing Data *TD* with Precision of 0.627 but Recall of only 0.143, although z* was chosen so as to maximize Precision subject to Recall being greater than or equal to 0.6 based on the Learning Data *LD*. Algorithm VII also works poorly but in a reversed manner with Precision of only 0.378 but Recall of 0.720. Algorithms I, III, IV and V perform very well for FGC-RP despite the fact that those algorithms were constructed based on

FGC-ALL. Algorithm I is quite valid also for FGC-PZ. For all other occasions, the algorithms built based on FGC-ALL perform less satisfactorily for the subcategories, which may be natural.

Table 5.6.3 Number of Smartphone Applications Satisfying or Not Satisfying End Condition

Category / Truth		I_{end}–FGC-ALL		I_{end}–FGC-RP		I_{end}–FGC-CAS		I_{end}–FGC-PZ		I_{end}–FGC-ACT	
Algorithms		1	0	1	0	1	0	1	0	1	0
		329		49		49		48		41	
			1144		92		196		158		146
Algorithm I	*Logistic Regression*	201		36		29		34		21	
			117		11		23		14		18
Algorithm II	*Decision Tree*	204		11		13		12		5	
			177		26		64		37		33
Algorithm III	*Average Model*	198		36		26		27		27	
			108		12		24		10		18
Algorithm IV	*Max Model*	200		36		29		30		25	
			114		14		23		12		20
Algorithm V	*Min Model*	198		36		29		25		26	
			143		13		29		12		25
Algorithm VI	*And Model*	47		7		7		10		2	
			28		4		5		2		5
Algorithm VII	*Or Model*	237		40		35		36		24	
			390		33		82		49		46

Table 5.6.4 Precision and Recall Resulting from Table 5.6.3

Algorithms		I_{end}–FGC-ALL		I_{end}–FGC-RP		I_{end}–FGC-CAS		I_{end}–FGC-PZ		I_{end}–FGC-ACT	
		Precision	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*
Algorithm I	*Logistic Regression*	0.632		0.766		0.558		0.708		0.538	
			0.611		0.735		0.592		0.708		0.512
Algorithm II	*Decision Tree*	0.535		0.297		0.169		0.245		0.132	
			0.620		0.224		0.265		0.250		0.122
Algorithm III	*Average Model*	0.647		0.750		0.520		0.730		0.600	
			0.602		0.735		0.531		0.563		0.659
Algorithm IV	*Max Model*	0.637		0.720		0.558		0.714		0.556	
			0.608		0.735		0.592		0.625		0.610
Algorithm V	*Min Model*	0.581		0.735		0.500		0.676		0.510	
			0.602		0.735		0.592		0.521		0.634
Algorithm VI	*And Model*	0.627		0.636		0.583		0.833		0.286	
			0.143		0.143		0.143		0.208		0.049
Algorithm VII	*Or Model*	0.378		0.548		0.299		0.424		0.343	
			0.720		0.816		0.714		0.750		0.585

One may expect that the segmentation algorithms built on individual subcategories would work better because the variability of individual subcategories would be less than that of FGC-ALL. In order to examine this hypothesis, we repeat the

development procedure for Algorithms I through VII again for FGC-RP. Table 5.6.5 reveals that the segmentation levels of the seven algorithms obtained for FGC-RP are much severer than those for FGC-ALL given in Table 5.6.2. Corresponding to Tables 5.6.3 and 5.6.4, the number of smartphone applications satisfying or not satisfying End Condition and the resulting Precisions and Recalls are summarized in Tables 5.6.6 and 5.6.7, respectively. In both Precision and Recall, the seven algorithms work much better for FGC-RP than for FGC-ALL, as expected.

Table 5.6.5 The Optimal Segmentation Level z* for FGC-RP

Algorithms		FGC-ALL
		z^*
Algorithm I	*Logistic Regression*	0.970
Algorithm II	*Decision Tree*	0.840
Algorithm III	*Average Model*	0.910
Algorithm IV	*Max Model*	0.970
Algorithm V	*Min Model*	0.840
Algorithm VI	*And Model*	0.970
Algorithm VII	*Or Model*	0.840

Table 5.6.6 Number of Smartphone Applications Satisfying or Not Satisfying End Condition for FGC-RP

Category / Algorithms	Truth	I_{end}–FGC-RP	
		1	0
		49	92
Algorithm I	*Logistic Regression*	34	6
Algorithm II	*Decision Tree*	32	13
Algorithm III	*Average Model*	34	15
Algorithm IV	*Max Model*	42	21
Algorithm V	*Min Model*	31	18
Algorithm VI	*And Model*	29	2
Algorithm VII	*Or Model*	37	17

Table 5.6.7 Precision and Recall Resulting from Table 5.6.5

Algorithms		I_{end}–FGC-RP	
		Precision	*Recall*
Algorithm I	*Logistic Regression*	0.850	0.694
Algorithm II	*Decision Tree*	0.711	0.653
Algorithm III	*Average Model*	0.829	0.694
Algorithm IV	*Max Model*	0.667	0.857
Algorithm V	*Min Model*	0.886	0.633
Algorithm VI	*And Model*	0.935	0.592
Algorithm VII	*Or Model*	0.685	0.755

It turns out that FGC-RP contains 141 applications. Since both Algorithm I and Algorithm II have shown excellent performance, and they allow us to explore a set of influential variables for making the judgments, we now attempt to derive some business implications useful for application developers based on Algorithm I and Algorithm II.

The results of LR are summarized in Table 5.6.8, where the estimated values of the coefficients of the explanatory variables with significance level of 0.001 are listed. Table 5.6.9 exhibits the results of DT, where the top 6 leaf nodes along with respective branching rules are listed in the order of precision.

Table 5.6.8 Estimated Values of Coefficients of Explanatory Variables in LR

variable	coefficient	Std. Error	z value	Level of significance
Intercept	-5.075	0.320	-15.845	***
X60	12.222	2.188	5.586	***
X61	12.019	1.811	6.635	***
X88	10.146	1.576	6.440	***
X58	7.546	1.266	5.959	***
X87	6.858	1.310	5.236	***
X95	6.598	1.196	5.515	***
X18	6.024	1.734	3.474	***
X1	5.147	0.917	5.615	***
X55	4.856	0.998	4.864	***
X67	4.463	1.128	3.958	***
X51	4.421	1.281	3.451	***
X20	4.392	0.929	4.730	***
X21	4.237	1.120	3.784	***
X92	4.074	0.911	4.474	***
X80	3.714	0.745	4.984	***
X10	3.608	0.708	5.093	***
X9	2.142	0.620	3.455	***
X11	-3.377	0.840	-4.019	***
X57	-4.561	1.238	-3.683	***
X12	-6.162	0.965	-6.388	***
X83	-6.337	1.143	-5.546	***
X63	-6.395	1.848	-3.461	***
X53	-7.637	2.118	-3.605	***
X49	-7.802	1.356	-5.755	***
X62	-9.402	2.012	-4.674	***
X94	-10.462	1.385	-7.554	***
X56	-10.904	1.757	-6.207	***
X77	-12.795	1.749	-7.317	***

Level of significance: '***' 0.001

Table 5.6.9 Branching Rules for Leading Leaf Nodes in the Order of Precision

Leaf_Node	N(Leaf)	Iend-SP(Leaf)	Rule1	Rule2	Rule3	Cum of N(L)	Cum of Iend-SP(L)	Precision	Recall
L1	47	32	X55 >= 0.01	X88 >= 0.04		47	32	0.681	0.653
L2	5	3	X55 < 0.01	X10 >= 0.4	X9 >= 0.07	52	35	0.673	0.714
L3	3	2	X55 >= 0.01	X88 < 0.04	X94 >= 0.1	55	37	0.673	0.755
L4	15	5	X55 >= 0.01	X88 < 0.04	X94 < 0.1	70	42	0.600	0.857
L5	11	2	X55 < 0.01	X10 >= 0.4	X9 < 0.07	81	44	0.543	0.898
L6	60	5	X55 < 0.01	X10 < 0.4		141	49	0.346	1.000

One realizes that the following 4 variables are chosen with importance in both LR and DT, where the definitions are given prior to the standardization in (5.6.3)

X9 $= Act\text{-}Count(a, male, \text{late}20's, t)$: the sum of the number of activations of $a \in A(T)$ in month t over $d \in D_{male,late20's}(T)$

X10 = $Act\text{-}Count(a, male, 30's, t)$: the sum of the number of activations of $a \in A(T)$ in month t over $d \in D_{male,30's}(T)$

X55 $= Act\text{-}Count\text{-}Change(a, male, 10's, t, t+2) = \frac{Act-Count(a,male,10's,t+2)}{\max\{Act-Count(a,\text{male},10's,t),\ 1\}}$

X88 = $Possess\text{-}Count\text{-}Change(a, female, 30's, t, t+2) = \frac{Possess-Count(a,female,30`s,t+2)}{\max\{Possess-Count(a,female,30`s,t),\ 1\}}$

Since the estimated values of X55 and X88 in Table 5.6.8 are positive, and since they appear in the branching conditions for Leaf Node 1 in Table 5.6.9, one may conclude that higher the values of X55 and X88 are, more likely the underlying smartphone application in FGC-RP would satisfy the sustainable popularity condition. This implies that the marketing campaign may be focused on male 10's and female 30's. In Figure 5.6.1, the cumulative number of smartphone applications in FGC-RP is plotted as a function of X55 or X88. One realizes that, for both male 10's and female 30's, about 50 % of smartphone applications in FGC-RP would not be activated again after the first use.

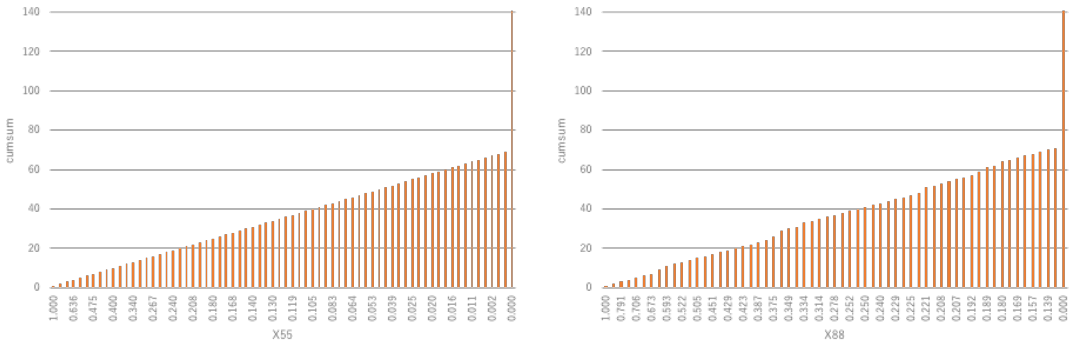


Figure 5.6.1 Cumulative Number of Smartphone Applications as a Function of X55 or X88

The branching rules for Leaf Node 2 in Table 5.6.9 reveal that even when X55 is less than or equal to 0.01, this fact can be compensated by the larger values of X9 and X10. This indicates that the marketing efforts focused on male late 20's through 30's may be also effective. The counterparts of Figure 5.6.1 for X9 and X10 are depicted in Figure 5.6.2. For male late 20's, about 63 % of smartphone applications in FGC-RP would not be activated again after the first use, while this figure for male 30's is about 48 %.

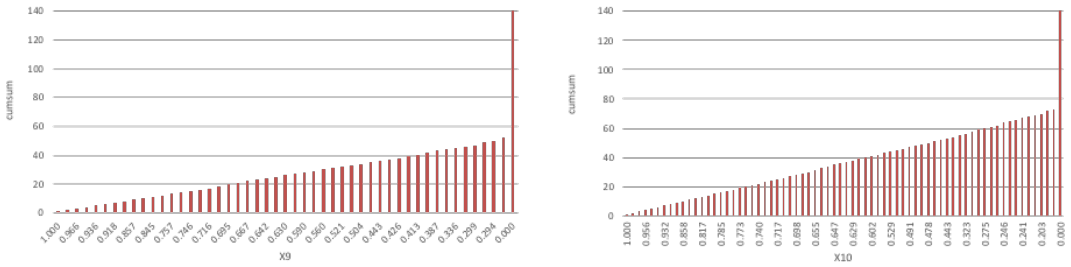


Figure 5.6.2 Cumulative Number of Smartphone Applications as a Function of X9 or X10

These observations have important business implications concerning how to organize the advertising campaign and the like, and would be quite useful for software development companies.

Chapter 6

Concluding Remarks

From the point of view of marketing analysis, the peculiarity of smartphone applications in comparison with ordinary products can be found in that they may be installed and uninstalled dynamically over time by customers in a repeated manner. The difficulty of acquiring such dynamic data from the market has been impeding the study of usages of smartphone applications by individual users. Recently, a Japanese software development company named Fuller, Inc. has overcome this challenging problem, successfully obtaining actual data concerning installments and un-installments of smartphone applications via the legitimate agreement between the company and the customers of this application.

Using the massive real data obtained from Fuller Inc., this thesis introduces the concept of "sustainable popularity ranking" for the first time in the literature, where, with $t_0(a)$ defined as the month in which application 'a' is introduced into the market, application 'a' is said to be with sustainable popularity ranking associated with (M, m_0, m_1) if it is ranked within top M by Google Play at least once during the first m_0 months since $t_0(a) + 1$, and is still ranked within top M at least once during the m_0 months after $t_0(a) + 1 + m_1$.

Using Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), Neural Network (NNet) and Naïve Bayes Classifier (NBC) provided by the R language, two combined segmentation algorithms are developed for identifying smartphone applications with the property of sustainable popularity ranking. The two algorithms are applied to massive real data obtained from

Fuller, Inc., achieving the high levels of Recall, Precision and Accuracy, thereby yielding useful business implications of interest to software development companies.

In order to identify the impact of individual components of the application profile vector on sustainable popularity ranking more deeply, five different algorithms are developed using only LR and DT. With focus on Japanese free games, some key factors are revealed that play a significant role in the segmentation algorithms. One finds that it may be effective to focus the marketing efforts on both male 10`s and female 30`s for Japanese free games in the category of Role Play (FGC-RP). It is also found that the marketing efforts focused on male late 20's through 30's may be effective for FGC-RP. Among customers in male 10's, male late 20's through 30's and female 30's, around 50 % of smartphone applications in FGC-RP would not be activated again after the first use. These findings are subtle and useful for software development companies, demonstrating the potential usefulness of the algorithms developed in this study.

This study is, however, still in its infancy and much more work should be done. As a limitation, it was not possible for us to identify and avoid some of the bias that could have affect the outcome if this study. For example, it is necessary to conduct the sensitivity analysis in terms of m_0, m_1 and M to see how the performance results, as well as the resulting business implications, would be affected as the values of m_0, m_1 and M change. It is also of interest to examine whether or not smartphone applications with sustainable popularity ranking could be characteristically different from those applications satisfying only the end condition associated with sustainable popularity ranking. Furthermore, more extensive efforts should be made so as to extract

business implications from the performance results of the segmentation algorithms. These studies are in progress and will be reported elsewhere.

Bibliography

AppBrain. (2016). Category statistics table. Retrieve from http://www.appbrain.com/stats/android-market-app-categories

AppBrain. (2019). Category statistics table. Retrieve from http://www.appbrain.com/stats/android-market-app-categories

Buja, A., & Lee, Y.-S. (2001). Data mining criteria for tree-based regression and classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, August 26-29, 2001. 27–36. doi: 10.1145/502512.502522

Chang, T., Qi, L., Enhong, C., & Hui, X. (2012). Prediction for Mobile Application Usage Patterns. In *Nokia MDC Work*, Nokia MDC Workshop, 4.

eMarketer. (2016). 2 Billion Consumers Worldwide to Get Smart (phones) by 2016. Retrieve from http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694.

eMarketer. (2016). Survey on Smartphone user penetration in Japan. Retrieve from http://www.emarketer.com/Article/Smartphone-Use-Japan-Makes-Steady-Gains/1010226.

EMC Education Services (2015). *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*. John Wiley and Sons. 432p. ISBN: 978-1-118-87613-8

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. (2010). Diversity in smartphone usage. In *Proc. 8th Int. Conf. Mob. Syst. Appl. Serv. - MobiSys '10*, 179-195.

Google (2016). Google Play website. Retrieve from https://play.google.com/

Hosmer, Jr., D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Introduction to the Logistic Regression Model, in Applied Logistic Regression.* Third Edition, John Wiley & Sons, Inc., Hoboken, NJ. doi:10.1002/9781118548387.ch1

Jelinek F. (2005). Language Modeling Experiments with Random Forests. In: *Matoušek V., Mautner P., Pavelka T. (eds) Text, Speech and Dialogue. TSD 2005. Lecture Notes in Computer Science*, *3658*. Springer, Berlin, Heidelberg.

Joachims, T. (2002). Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithm. In *The Springer International Series in Engineering and Computer Science*, *668,* 35-44.

Lin, Y., Wang, J., & Zou R. (2015). An Improved Naïve Bayes Classifier Method in Public Opinion Analysis. In *Wong W. (eds) Proceedings of the 4th International Conference on Computer Engineering and Networks. Lecture Notes in Electrical Engineering*, *355*. Springer, Cham. doi: 10.1007/978-3-319-11104-9_26.

Minh, T., Do, T., Blom, J., & Gatica-Perez, D. (2011). Smartphone Usage in the Wild : a Large-Scale Analysis of Applications and Context. In *Proc. ICMI '11, 13th Int. Conf. multimodal interfaces*, 353-360.

Mizuno, M., Saji, A., Sumita, U., & Suzuki, H. (2008). Optimal Threshold Analysis of Segmentation Methods for Identifying Target Customers. *European Journal of*

Operations Research, 186, Issue 1, 1-442.

Mizusawa, S., Sumita, U., & Takano, M. (2015). Prediction of the Number of Downloads Of Smartphone Applications Based on the Markov Chain Approach. In *Int. J. Bus. Inf.*, *10*(1), 1-23.

Perera, U., Dewi, C., Sari, M., & Sumita, U. (2014). Analysis of Interrelationships Among Application Popularity, Application Stability, and Potential Risk in e-WOM. In *Int. J. Bus. Inf.*, *9*(3), 235-272.

Perera, U., Shigeno, M., Sumita, U., & Yamamoto, Y. (2016). Development of statistical approach for estimating key competitive performance measures of smartphone applications. In *Advances in ICT for Emerging Regions (ICTer), 2016 Sixteenth International Conference on IEEE*, Negombo, Sri Lanka, September 1-3, 2016. 266-273.

Piovoso, M. J., & Owens, A.J. (1991). Neural network process control. In *Proceedings of the Conference on Analysis of Neural Network Applications*, ANNA '91, Larry K. Barrett (Ed.), ACM, New York, USA, 84-94. doi: http://dx.doi.org/10.1145/ 106965.105256.

Statista (2019). Number of smartphone users worldwide from 2016 to 2021. Retrieve from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Smaato (2018). Japan: Mobile Usage and Spending by the Numbers. Retrieve from https://www.smaato.com/blog/japan-mobile-usage-and-spending-by-the-numbers

Sumita, U. (2018). Important Analytical Tools for Big Data Analytics, Part I: Time

Series Analysis and Segmentation Algorithms. In *Introduction to Big Data Analytics,* Spring, 2018.

Sumita, U. (2018). Important Analytical Tools for Big Data Analytics, Part II: Decision Tree. In *Introduction to Big Data Analytics,* Spring, 2018.

Sumita, U. (2018). Important Analytical Tools for Big Data Analytics Part III: Neural Network. In *Introduction to Big Data Analytics,* Spring, 2018.

Tongaonkar, A., Dai, S., Nucci, A., & Song, D. (2013). Understanding Mobile App Usage Patterns Using In-App Advertisements. In *Passive and Active Measurement*, 63-72.

Xu, Q., Mao, Z. M., Arbor, A., Erman, Park, J. F., Gerber, A., Pang, J., & Venkataraman, S. (2011). Identifying Diverse Usage Behaviors of Smartphone Apps. In *Proc. 2011 ACM SIGCOMM Conf. Internet Meas. Conf.*, 329-344.