A SELF-LEARNING AUDIO PLAYER THAT USES A ROUGH SET AND NEURAL NET HYBRID APPROACH

by

Hongming Zuo

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science (MSc) in Computational Sciences

The School of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE

Laurentian Université/Université Laurentienne
School of Graduate Studies/École des études supérieures

Title of Thesis Titre de la thèse	A SELF-LEARNING AUDIO PLAYER THAT USES A ROUGH SET AND NEURAL NET HYBRID APPROACH
Name of Candidate Nom du candidat	Zuo, Hongming
Degree Diplôme	Master of Science

Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance	October 04, 2013

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Julia Johnson
(Supervisor/Directrice de thèse)

Dr. Ralf Meyer
(Committee member/Membre du comité)

Prof. Aaron Langille
(Committee member/Membre du comité)

Dr. Khaled Mahmud
(External Examiner/Examinateur externe)

Approved for the School of Graduate Studies
Approuvé pour l'École des études supérieures
Dr. David Lesbarrères
M. David Lesbarrères
Director, School of Graduate Studies
Directeur, École des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

A self-learning Audio Player was built to learn users habits by analyzing operations the user does when listening to music. The self-learning component is intended to provide a better music experience for the user by generating a special playlist based on the prediction of users favorite songs. The rough set core characteristics are used throughout the learning process to capture the dynamics of changing user interactions with the audio player. The engine is evaluated by simulation data. The simulation process ensures the data contain specific predetermined patterns. Evaluation results show the predictive power and stability of the hybrid engine for learning a users habits and the increased intelligence achieved by combining rough sets and NN when compared with using NN by itself.

Keywords

Artificial Neural Network, Rough Set, self-learning, hybridization

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Prof. Julia Johnson for the continuous support of my master study and research. Her guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank my committee members, Prof. Ralf Meyer and Prof. Aaron Langille for their insightful comments and hard questions.

Last but not the least, I owe sincere and earnest thankfulness to my family: my parents Jianfeng Zuo and Tonghui Li, for giving birth to me at the first place and supporting me spiritually throughout my life.

Table of Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

1 Introduction

1.1 Background

An audio player is popular software in every day life. Most people are using it to enjoy music when they are jogging, reading, resting and so on. There are a lot of different types of audio players, for example, Windows Media Player and iTunes. Most of them have just a basic function, like playing songs or shuffling song lists.

When we will use the short form APP, we are referring specifically to a digital iPhone application though apps are used for all sorts of platforms other than digital iPhone applications, and we will use iOS to mean iPhone operating system. The Apple App store is a digital application distribution platform for iOS, developed and maintained by Apple Inc. People can develop their own applications, and publish them in the APP store. As of February 10, 2012, there are more than 700,000 third-party apps officially available on the APP store. Compared with the other digital platforms, the iOS APP store is the most popular digital application store. As in May 15, 2013, downloads from the APP store reached 50 billion. The goal of the thesis is to develop and publish an audio player that learns about the user's preferences in the APP store. It can self-improve and self-learn based on an artificial neural network.

An artificial neural network (ANN), usually called neural network (NN), is a mathematical model or computational model that is inspired by the structure and/or functional aspects of biological neural networks. It involves machine learning. In order to learn by itself, the machine has to have an adaptive system. Moreover, the main point of neural networks for machine learning is the notion of a perceptron (Neuron + Weight training).

The time that is used in NN's weight training will be reduced when the number of weights to train is decreased. The number of weights is the same as the number of condition attributes in an information table, and Rough Set (RS) is a paradigm that can be used to reduce the number of condition attributes. Rough data are characterized by imprecision, inconsistency and incompleteness. Hybrid approaches that combine NN and Rough Set have been explored in the literature. Rough Set method can be used to generate rules from rough data. The rules will be very useful to analyze data about user preferences when they play audio especially when used in conjunction with NN. An objective toward achieving the goal of the thesis is to develop a RS and NN hybrid system that learns user's preferences in a music player.

1.2 The current situation

iTunes is a product developed by Apple to play music. iTunes has a inventive function called Genius. This function can recommend playlists (a list of songs), provide a mixture

of songs that go great together chosen from different libraries, and suggest songs that the user may like. The mechanism by which iTunes Genius works is proprietary to Apple. Researchers are studying Genius, trying out Genius on a test collection of music, and analyzing how it may be working. Their conclusion[1] is, first of all that Genius performs well at detecting acoustically similar songs, and secondly that its recommendation is derived from a purely content-based system. By content-based system, they mean that the songs are compared by descriptors developed by musicologists (people who studies music) to judge whether two songs sounds similar. Those researchers contrast the content-based approaches with the meta- data approaches by which they mean the information about music such as artist, album and genre.

iTunes records detailed information about songs the user has played including how many times the song has been played. It seems as though iTunes Genius [1][2] recommends the playlist by analyzing such detailed information that reflect the habits of users, like which genre of music the user usually listens to, which artist is the user's favorite and which album the user tends to play. It is a very interesting area in AI to build software that can be open to the research community to show how user habits can be learned to make a solution different than iTunes Genius.

1.3 Thesis Objectives and Novelty

The main objective of this thesis is to analyze music with respect to the user's operation

on it such as how many times the song has been skipped and how many times the song has been picked. The first novelty of our approach is to develop a rough set and neural network hybrid system that learns user's preferences in the music player. While others have used previously implemented RS and NN engines as third-part software, we implement the RS and NN engine by developing our own software. That is, we do the hybridization at the code level. The expected advantage is avoidance of the manual step between the Rough Set engine and the NN, that was required in the previous approaches[3][4].(See Figure 1.1 (a)) Figure 1.1 (b) illustrated our integrated approach in which we use rough set's core characteristics to guide NN's learning process in code.

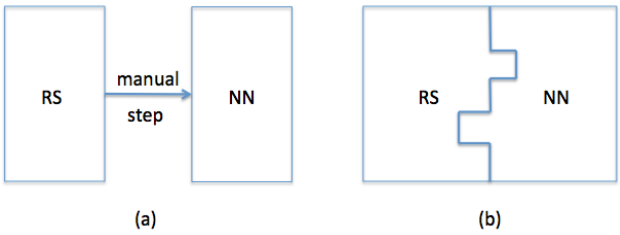


Figure 1.1: Comparison of previous and our approach to RS/NN hybridization

In previous work, RS core characteristics were used to initialize the weight of inputs to the NN. We tried this approach and found that it does not offer much advantage in improving the learning speed of the NN. However, in our work, we are using the important attributes to guide weight training throughout the training process not just at initialization to make sure the insignificant attributes will not have any influence on the learning process. The novelty lies in development of dynamic weight training because an attribute may become significant as people use this system while earlier it was not significant. Conversely, a significant attribute may no longer be significant.

Chapter 2

2 Previous Work

2.1 Rough Sets

2.1.1 Fundamentals

Rough Set Theory (RST)[5] provides an idea to deal with imprecision in data. RST can also be used to extract rules from concise data. In order to understand RST, we need to know about the following concepts. [6]

Information system framework: Adopting common notation in the field, we assume U to consist of objects in the universe, and A to consist of features of those objects. Then, $I = (U, A)$ is an information system. For every $a \in A$, there are $a: U \to V_a$. The V_a is the set of values that attribute a may assume.

Equivalence relation: For every $P \subseteq A$, there is an equivalence relation $IND(P)$. It is called the P-indiscernibility relation, defined as follows:

$$IND(P) = \{(x, y) \in U^2 | \forall a \in P, a(x) = a(y)\}$$

Lower and Upper approximation: In RST, lower approximation and upper approximation are very important concepts.

X denotes an object set to be approximated where $X \subseteq U$. P denotes an attribute subset

of A. $[x]_p$ denotes an equivalence class of P-indiscernibility relation. Lower and upper approximations are defined as follows:

$$\underline{P}X = \{x \big| [x]_p \subseteq X\} \ (lower\ approximation)$$

$$\overline{P}X = \{x \big| [x]_p \cap X \neq \emptyset\} \ (upper\ approximation)$$

The P-lower approximation is also called the positive region. It is the union of all P-indiscernibility classes $[x]_p$, which are subset of X. The P-upper approximation also called the negative region is a set of all P-indiscernibility classes $[x]_p$ that intersect with X.

Boundary region: The boundary region is the difference between upper approximation and lower approximation.

$$BND_p(X) = \overline{P}X - \underline{P}X$$

If $BND_p(X)$ is empty, that means $\overline{P}X = \underline{P}X$. Concept X is said to be P-definable. The pair of crisp sets $\langle \underline{P}X, \overline{P}X \rangle$ is a rough set with respect P [5].

Reduct and Core: For any attribute $a \in A$, a is dispensable in the set A if $IND(A) = IND(A - a)$. If we eliminate all dispensable attributes from A, we get P as the reduct set of A, denoted by $RED(A)$(using notation in [4]).

$$RED(A) = \{R: R \subset A, IND(R) = IND(A)\}$$

The intersection of all the reducts of A is called the core.[4]

$$CORE(A) = \cap\, RED(A)$$

All attributes in the core are indispensable for approximating X. They cannot be removed from A without changing the original classification.

Precision of Rough Set: There are a series of measures for the uncertainty contained in information [7]. The accuracy of the rough set approximation can be formulated as follows:

$$a_p(X) = \frac{|\underline{P}X|}{|\overline{P}X|}$$

This provides a measure of the closeness between upper approximation and lower approximation. If the approximation is perfect, then $a_p(X) = 1$. If the number of objects in the lower approximation is 0, then $a_p(X) = 0$.

2.1.2 Rough Set Algorithm

Even today, most of rough set engines in general use the original algorithm given by Pawlak [5], refined by Wong and Ziarko[8], refined further by many others. Currently, an area of active research into rough sets focuses on hybrid approaches [9][10][11][12] that integrate rough set theory with other paradigms for dealing with uncertainty. Here the original algorithm will be described which generates a set of decision rules from data.

In this thesis, the major rough set algorithm is RS1 algorithm by Wong[13][14]. The basic idea of this algorithm can be described as follows. For concept X, RS1 evaluates each attribute subset P by comparing $a_p(X)$ for P a subset of condition attributes.

RS1 Algorithm

Generate decision rules from Decision table T

Let $C = \{C_1, C_2, \ldots C_j\}$ is the set of condition attributes

Let X represents the set of elements in T
P is a subset of condition attributes
Let $D = \{D_1, D_2, D_3, \ldots D_n\}$ be the partition of decision table based on decision attribute values, and $\underline{P}D_i$ is lower approximation of concept D_i

for D_1 to D_n **do**

while P cannot increase any more

begin with $P = \{C_1\}$ to $P = \{C_j\}$

compare each P to find which $a_p(D_i)$ is greatest.
Assume when $P = \{C_1\}$, then $a_p(D_i)$ is greatest.
The deterministic rules can be generated from all elements X_{sub} in $\underline{P}D_i$.
The non-deterministic rules can be generated from $BND_p(D_i)$
Then, pick $p = \{C_1\}$ as root, and put X_{sub} out of X

Begin with $P = \{C_1, C_2\}$ to $P = \{C_1, C_j\}$ do above again

…
…
Begin with $P = \{C_1, C_2, C_3 \ldots C_k\}$ to $P = \{C_1, C_2, C_3 \ldots C_l\}$ do above again
….
….

End while

Reset X and P as beginning.

end for

The algorithm will pick the P for which $a_p(X)$ is greatest. Then, the elements in $\underline{P}X$ can be used to generate deterministic decision rules, and the elements in $\overline{P}X$ but not in $\underline{P}X$ can be used to generate nondeterministic decision rules. The pseudo-code of RS1 algorithm follows. The pseudo-code is based on the same idea as the learning algorithm in [8], but the notation and description has been changed to make it easier to understand.

In the project, the algorithm will generate both nondeterministic and deterministic rules. However, the rules generated by the RS1 inductive learning algorithm are not used to train weights for the ANN. It is the capability of the inductive learning algorithm to find a minimal set of attributes from an available set of condition attributes that was used to assist with weight training. The generation of if-then rules together with a measure of the strength of the rules may find use in future work regarding this project. In chapter 4, modification to make the original algorithm suitable to the situation at hand will be described in detail.

2.2 Artificial Neural Network

The hybrid approach used in this thesis involves an artificial neural network. A neural network can be defined as a model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called neurons. ANN results in a machine that can learn by itself as a human does.

An artificial neural network (ANN), usually called neural network (NN), is a mathematical model or computational model that is inspired by the structure and/or functional aspects of biological neural networks. It involves machine learning. To learn by itself, the software has an adaptive system based on ANN. The main point of neural network is the weight learning, which is the training of weights for inputs to the ANN by using sample data.

There are several types of neural networks and learning algorithms. According to Zhang [15], one hidden layer neural network is sufficient to deal with any complex system. The different types of learning algorithms include the gradient descent method, the conjugate gradient method, the Gauss-Newton algorithm [16], and the Levenberg-Marquardt (LM) algorithm [17]. Different types of algorithms influence the speed and precision of the training process. The LM algorithm is the most popular, because it uses the Gauss-Newton method blended with gradient descent which leads to fast convergence [18].

2.2.1 Single-layer Neural network

An artificial neuron is attempting to do the same work as basic information processing typically done by humans (Figure 2.1). For example, the neuron of NN corresponds with the soma of a human brain, the input of NN to the dendrite of human brain, the weight of NN to the synapse of the human brain and the output of an NN to the axon of human brain.

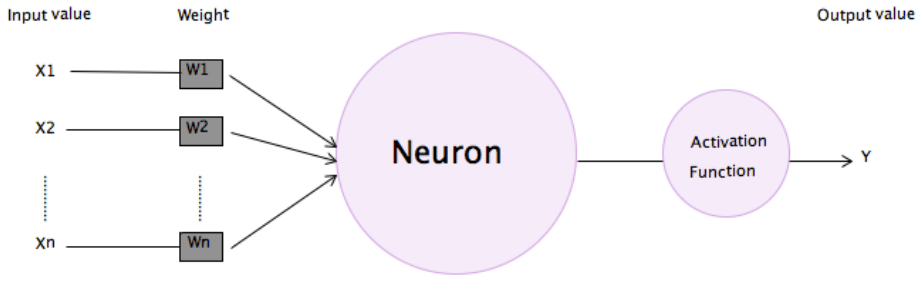


Figure 2.1: Single-layer neural network

The neurons will computer the weighted sum of the input signals and compare the result with a threshold value θ. If the input is smaller than the threshold value, the output will be -1, otherwise, it will be +1. So, the formulae are (X is input, and Y is output) as follows [19]:

$$X = \sum_{i=1}^{n} x_i w_i$$

$$Y = sign[\sum_{i=1}^{n} x_i w_i - \theta]$$

In the output formulae, the 'sign' is a function called the activation function, and there are many other activation functions like Step, Sigmoid, and Linear. In a single-layer neural network model, we use the Step activation function to get the result.

The Step and Sign activation function are also called hard limit functions. They are used in classification and pattern recognition task.

$$Y^{step} = \begin{cases} 1, if\ X \geq 0 \\ 0, if\ X < 0 \end{cases} \qquad Y^{sign} = \begin{cases} +1, if\ X \geq 0 \\ -1, if\ X < 0 \end{cases}$$

The Sigmoid activation function is used in the back-propagation network that will be shown in subsection 2.3.2 to follow.

$$Y^{sigmoid} = \frac{1}{1 + e^{-X}}$$

The Linear activation function gives the output equal to the neuron-weighted input. Neurons with the linear function are usually used for linear approximation.

$$Y^{linear} = X$$

The algorithm for training a single-layer neural network concerns a perceptron that was introduced by Frank Rosenblatt [20]. Refer to Figure 2.2 that shows the learning process.

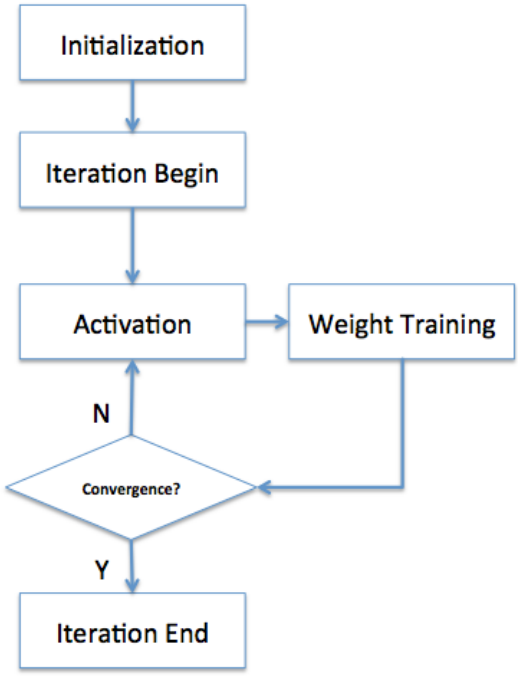


Figure 2.2: The perceptron algorithm

Step 1: Initialization

We initialize the threshold θ and the weights $w_1, w_2 \ldots w_n$ for each input. We set each w_i to a real value in the range $[-0.5, 0.5]$.

Step 2: Begin iteration

Step 3: Activation

Applying inputs $x_1(p), x_2(p) \ldots x_n(p)$ to an activation function and get the actual output Y (p). P is the iteration number.

$$Y(p) = step[\sum_{i=1}^{n} x_i(p)w_i(p) - \theta]$$

The n is the number of input values; step is one kind of activation function as:

$$Y^{step} = \begin{cases} 1, if\ X \geq 0 \\ 0, if\ X < 0 \end{cases}$$

Step 4: Weight Training

Calculate the error between the desired output $Y_d(p)$ and the actual output $Y(p)$.

$$e(p) = Y_d(p) - Y(p)$$

Then, the system will use error to train the weigh as following formulas:

$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p)$$

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

Where α is learning rate, it is a variable that defines the degree of weight correction. When α is large, the weight correction range will be larger. When α is small, the weight correction range will be small. The next iteration will use the new weight.

Step 5: Check whether convergence has been reached

When the error is small enough, we treat that as system convergence. When convergence has been reached, we go to step 6, otherwise we go back to step 3. Iteration will continue until the system reaches convergence.

Step 6: End iteration

By using a singe-layer neural network, we can solve some simple problems like learning how to do basic logical operations. If the problem is complicated, we need a more complex neural network.

To conclude this section we would like to show single layer NN training for learning the meaning of logical operations that was previously done as part of a course project undertaken by the author of this thesis. This exposition illustrates clearly where a single layer NN breaks down in its learning capability.

The objective was the simulation of a simple neuron using a Java program. The perceptron (neuron + weight training) was tested by having it learn the meaning of the logical relations (*and*, or, *exclusive or*). It is possible using a simple NN to learn the meaning of *and* and *or* but not *exclusive or.* To learn the meaning of *exclusive or* a multilayer NN is required. Figure 2.3 gives insight into why this might be the case. Observe that the graphical representation of *and* and *or* requires one line to separate the true from false outcomes while *exclusive or* requires two lines to do this. In Figure 2.3, the black dots indicate the output is 1, and the white dots indicate the output is 0.

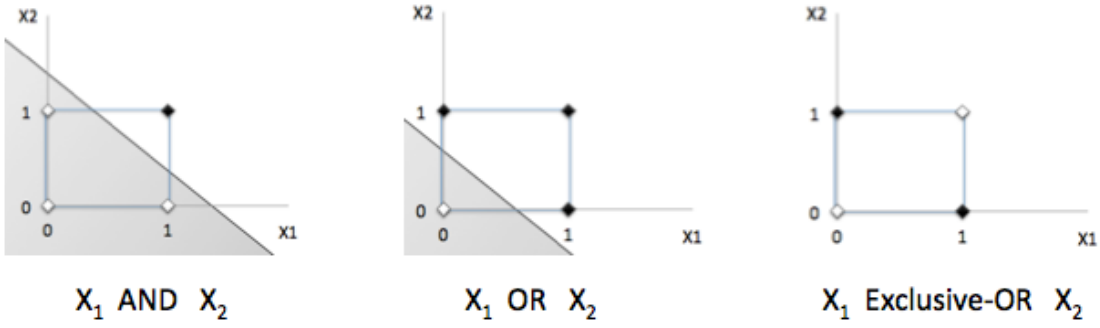


Figure 2.3: Two-dimensional plots of basic logical operation.

It was found that the learning rate is very important for the simple neural network. When the learning rate was set to 0.1, sometimes the program goes into an infinite loop. However, when the learning rate was set to 0.0001, the infinite loop problem was gone. The learning rate's function is to adjust the weight to a certain precision. If the learning rate is too large, the training of weights will not be successful. More successful weight training was obtained with a smaller learning rate.

2.2.2 Multilayer Neural network

As its name suggests, a multilayer neural network consists of multiple layers of neurons. Normally, there is an input layer, hidden layers and an output layer as illustrated in Figure 2.4.

In a multilayer neural network, the learning process is the same as single-layer neural network but a different algorithm is used for activation and weight training. In particular, the neurons in the hidden layer process inputs and give the output to every neuron in the output layer.

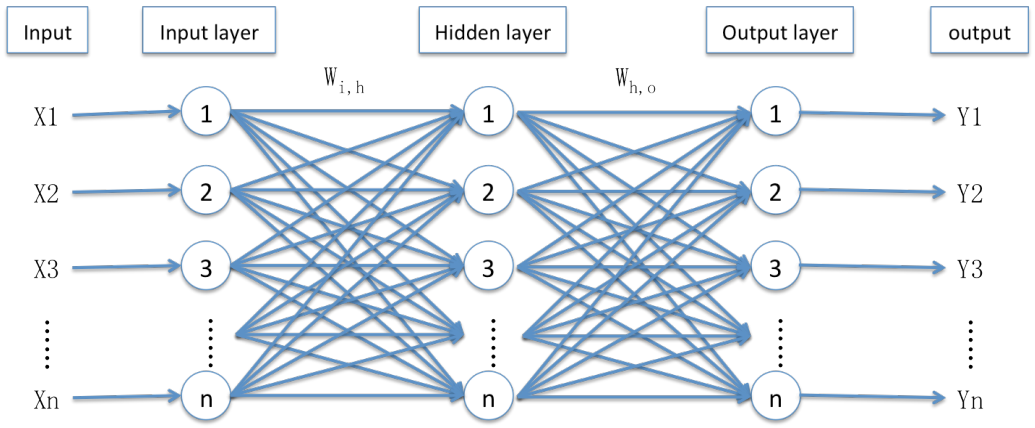


Figure 2.4: The multilayer neural network

In a multilayer neural network, the idea of the learning law is that of back-propagation. Figure 2.5 is like Figure 2.4, but the latter shows data flow. Referring to Figure 2.5, the input signals $x_1, x_2, \ldots, x_n$ flow into the system from left to right. After we get the output, we will get the error between the actual output and desired output. The error signal will return to the beginning as illustrated in Figure 2.5. Based on the error signal, the ANN trains itself.

We have known from the single-layer neural network that we train the weight by using the error. In multilayer neural network, there is a new concept named error gradient [21]. The error gradient is used for weight correction as in following formula:

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$$

where $\delta_k(p)$ is the error gradient at neuron k, and p means it is the p th iteration. The α is the learning rate. The $y_j(p)$ is the output value on neuron j. The error gradient is determined by a formula as follows:

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p)$$

$$e_k(p) = y_{d,k}(p) - y_k(p)$$

where $y_{d,k}(p)$ is the desired output of neuron k, and $y_k(p)$ is the actual output. The $e_k(p)$ is the error between them.

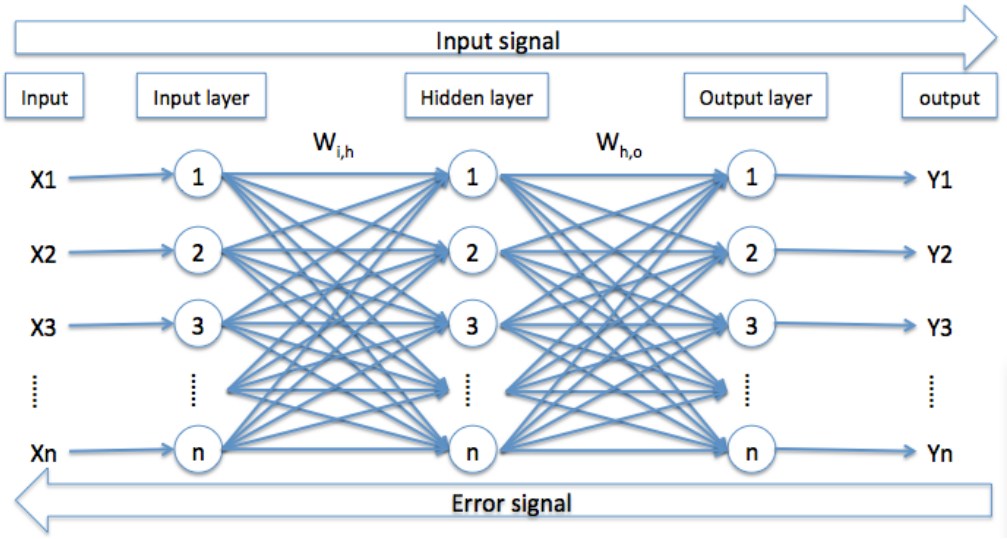


Figure 2.5: The back-propagation of multilayer neural network

The whole process is almost the same as single-layer neural network as illustrated in figure 2.2, except that multilayer neural network uses a more complicated way to process the error.

Step 1: Initialization

We will set the weights and threshold of each neuron to a random number in the neural network. The random number is randomly generated over a small range as follows [22]:

$$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i}\right)$$

The F_i denotes the total number of input values of neuron i in the network. Each neuron has its own initialization of its weights and threshold.

Step 2: Iteration begins

Step 3: Activation

We activate the multilayer neural network by using the inputs $x_1(p), x_2(p) \dots, x_n(p)$ and desired output $y_{d,1}(p), y_{d,2}(p), \dots, y_{d,n}(p)$. The p is the iteration number.

The hidden layer:

The actual outputs of each neuron are calculated, where $x_i(p)$ denotes the input from *i th* input, and $y_j(p)$ denote the output of *j th* neuron in hidden layer,

$w_{ij}(p)$ *denote the weigh for i th* input in $j\ th$ neuron in hidden layer

$$y_j(p) = sigmoid[\sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j]$$

The n is the number of inputs of neuron j in the hidden layer received from the input layer. Recall from Section 2.2.1, an explanation of why the activation function is changed when we move to multilayer NN. There we saw that the step function is capable of modeling the simple and/or logic connectives, but break down on exclusive-OR. In general, the sigmoid function is required to model some of the more complex problems. The $sigmoid$ is the sigmoid activation function as:

$$Y^{sigmoid} = \frac{1}{1 + e^{-X}}$$

The output layer:

The actual outputs of the neurons are calculated using the following formula, where $x_{jk}(p)$ denote the result from j th neuron of hidden layer to k th neuron of output layer, $y_k(p)$ denotes the output of k th neuron of output layer:

$$y_k(p) = sigmoid[\sum_{j=1}^{m} x_{jk}(p) \times w_{jk}(p) - \theta_k]$$

The m is the number of inputs of neurons in the output layer. Also, the m is the number of the output of hidden layer.

Step 4: Weight training

Train the weight of each neuron, do the weight correction. Because it is a back-propagation network, we start the training process from the output layer and move backwards to the input layer.

Firstly, we calculate using the back-propagation algorithm, the error gradient for the neurons from the output layer.[22]

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p)$$
$$e_k(p) = y_{d,k}(p) - y_k(p)$$

This is a two-part process:

1. Calculate the weight correction:

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$$

2. Update the weights at the output neurons based on their weight corrections:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

If p is the iteration number, then p+1 means the next iteration.

Secondly, we begin to deal with the error gradient for each neuron in the hidden layer:

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) \times w_{jk}(p)$$

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p)$$

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

The l denotes the number of output neurons.

Step 5: Check whether convergence has been reached

Each element from the training set inputs in the ANN will have errors between the desired output and the actual output. After iteration, if the sum of squared errors is less than one specific value, we treat the ANN as fully convergent. If it has reached convergence, we go to step 6, otherwise, we go back to step 3. The special value defines how accurate the neural network will be. The smaller this value is, the more accuracy the neural network will have.

Step 6: Iteration end

By using a multilayer neural network, we can solve some more complicated problems. The complexity of the problems to be solved determines how many neurons will be needed in the neural network. The more neurons the network has, the more complicated

the problem that can be solved. However, more neurons mean that it will take more time to train the neural network.

2.3 Hybrid system of neural networks and rough sets

There are a several approaches that have been used to combine rough set and neural network to solve problems. There is much research reporting on using a RS/NN hybrid system to solve specific problems.

In [3], the researchers apply a neural network and rough set method to solve the problem of credit scoring. In that paper, the rough set method is used to core the training set by which we mean the determination of significant attributes. If the important attributes are known, the calculation in NN can be reduced. Because the number of input attributes to the NN has been reduced, the cost of calculation in NN is also reduced. This approach was found to be very useful for credit scoring.

Referring to [4], the authors used the hybrid system of neural network and rough set to work on road safety performance indicators. In this paper, they also use the core function of the rough set method by using ROSETTA [23]. They use its core function to help the neural network to initialize weights. In general, the initial values of weights will play an important role in determining how effective the NN will be. Coring the training set is done in a separate step from NN training to get the significant attributes. Then, they initialize the values of the weights of significant inputs differently from the values of

weights of the other inputs. By using this method, they guide the network training, but in this thesis, we use a more integrated approach to RS-NN hybridization.

Referring to [11], the authors combine rough set and neural network. Their point in using rough set is to help in decreasing the calculation effort required for building the artificial neural network. The rough neural networks are constructed with three layers. The hidden layer and output layer consisted of the normal neurons as have previously been described. However, the input layer consists of rough neurons. This paper introduced a new hybrid network model, and the accuracy and stability of this model were shown to be good enough to solve a real life problem.

There are still a lot of papers [24][25] mentioning the use of a hybrid system of neural network and rough set to solve a variety of real life problems. They have different ways to combine rough set and neural network. The commonality among them is that they all use rough set to get the significant attributes, and then they use the significant attributes to improve the neural network resulting in reduction of the calculation effort of the neural network.

2.4 Discretization

Discretization is a process of transferring continuous models and equations to their corresponding discrete models and equations. The objective of this step is to convert the continuous data collected by the music player into discrete data compatible with both RS

method and NN. There are many discretization methods [26][27][28]. In this project, we use equal frequency binning discretization method [29], because compared to the other methods, this one costs less running time. The discretization being used divides the range into n intervals where the frequency of the objects is roughly the same in each interval. Bins are the sub ranges that divided the original set. In the equal frequency binning discretization method, the width of each bin depends on the size of the sample. The following is an example of equal frequency binning discretization method.

Example of equal frequency binning discretization method

There are 10 examples with the following values that need to be discretized.
7, 9, 14, 37, 67, 84, 86, 90, 92, 97
Suppose that we want to divide them into 5 bins. That means each bin has 2 elements.

7, 9	14, 37	67, 84	86, 90	92, 97

The boundary values of the bins are as follows.
$(9 + 14)/2 = 11.5 \quad (37 + 67)/2 = 52 \quad (84 + 86)/2 = 85 \quad (90 + 92)/2 = 91$
So, the bins value ranges will be as follows.
Bin 1: $[0, 11.5]$
Bin 2: $(11.5, 52]$
Bin 3: $(52, 85]$
Bin 4: $(85, 91]$
Bin 5: $(91, 97]$

There are two main parts of this project, the hybrid engine and the music player. The discretization is related to the hybrid engine. We now consider the music player that collects data for input into the hybrid engine.

2.5 Shuffle Algorithm

Generating shuffle playlist is a basic function of music player. Since the music player is

implemented as part of this project, we need to implement the shuffle algorithm. A shuffle algorithm is typically used to rearrange all elements in one data structure randomly.

The basic process of a shuffle algorithm is as follows [30]: Assuming that there are N elements in an array, first of all, generate a pseudo-random real number r $(0 \leq r \leq 1)$. A pseudo-random number differs from a true random number in that a computer generates a pseudo-random number whereas a truly random number occurs in a nature as an unpredictable phenomenon, like atmospheric noise.[31] A pseudo-random number r is more useful for software. Then, r is used as a multiplier, of the total number N. The resulting number i will be the new position of an element. In the basic process of shuffle, the elements are not swapped but only their associated index positions are reassigned. Collisions (an index may be chosen more than once) may occur frequently, because the new position may already be used. These collisions will reduce efficiency of the algorithm, especially common near the end.

There are some other shuffle algorithms that can avoid collisions. The strategy is to assign each element a random number that is in a larger range, and then sort the elements, according to their numbers in the larger space. These algorithms are very easy to implement, and computational complexity of these algorithms can be as low as O $(n \log n)$.[30] But, these algorithms required a lot of memory accesses, which can be more than the cost by collisions.

The original Fisher-Yates shuffle algorithm[32] used a better strategy to solve collisions. It generates a random number j between 1 and N, and then the element in j position of array will be picked up and put in a new array. After that, it generates a number from 1 to N-1, picks up an element that remains in the array and puts it into new array. It keeps doing this until all elements go into the new array. At termination, the new array is a random permutation of the original array. The modern algorithm avoids collisions and also has lower memory accesses than the Fisher-Yates original algorithm. The following sections outline the original and some variants of the Fisher-Yates shuffle algorithm.

2.5.1 Fisher-Yates original algorithm

This method was first designed by using pencil and paper in 1938 by Ronald A. Fisher and Frank Yates and, therefore it can be implemented on the computer using an algorithm. The pseudo-code of Fisher-Yates original algorithm is as follows.[32]

Fisher-Yates original algorithm

Shuffle and array $A_{original}$ of N elements. (Indices $0,1,2 \ldots N-1$):
1: $n = N$
2: **for** m = 0 to N-1 **do**
3: generate a random real number j ($0 \leq j \leq 1$)
4: the position $p = (j \ mod \ n)$
5: put a_p ($a_p \in A_{original}$) to b_m ($b_m \in B_{new}$), delete a_p from $A_{original}$
6: tag the elements of $A_{original}$ from 0 to N-2 by following same sequence.
7: $n = n - 1$
8: **end for**
return B_{new};

2.5.2 The Modern algorithm

This algorithm is the modern version of the Fisher-Yates original algorithm, designed for a computer.[33] The improvement of modern algorithm is that modern algorithm does not need to count remaining elements in the old array. That reduces computational complexity, and reduce the algorithm's time complexity to $O(n)$. The pseudo-code of modern algorithm is as follows.[33]

The modern algorithm

Shuffle and array A of N elements. (Indices $0,1,2 \ldots N-1$):
1: **for** i = N-1 to 1 **do**
2: generate a random real number j ($0 \leq j \leq 1$)
3: exchange a_i and a_j
4: **end for**
return A;

In this section, we introduced the main idea about rough sets, neural networks, discretization, shuffle algorithms and hybrid system of RS and NN. The neural network can learn from data. The rough set can core the condition attributes. By combining these two methods, we can have a hybrid approach to learn about music preferences. In the next section, we will present the statement of the problem that will be solved by combining the main ideas that were presented in this chapter.

Chapter 3

3 Problem Statement

The goal of thesis is to provide a self-learning Audio Player to learn a user's habits by analyzing operations the user does when listening to music on a hand held device. The self-learning component is intended to provide a better music experience for the user by generating a playlist specifically tailored to the user's music preferences.

The self-learning audio player will be evaluated using data that simulates how people listen to music. The simulated data follow some rules that express how a typical human being chooses and listens to music. The data are partitioned into training and testing sets. The audio player will be tested to see whether it can find the patterns that were intentionally introduced to the data. We know the desired result and our evaluation mechanism generates a measure of the distance between desired and predicted result.

The self-learning function is based on a Rough Set and Neural Net (NN) Hybrid approach. The hybrid engine will be designed to be portable not relying on third party software for the rough set engine or an off the shelf NN engine. The rough set core characteristics are used for reducing the number of attributes whose weights need to be trained, not only when initializing the NN, but throughout the learning process to capture the dynamics of changing user interactions with the audio player.

There were a number of major considerations in developing this project. Examples include design of software, design of Artificial Neural Network Engine and Rough Set Engine and the requirement for the portability of these engines. In this chapter we describe the major components of a hybrid approach to development of an adaptive playlist predictor.

3.1 Hybridization

It is obvious that there is a very important issue in constructing a neural network. It is about how to initialize the network weights and the number of iterations [3]. It will take a long time and a lot of iterations to run the neural network if we do not initialize a better weight than randomly generate the weigh for example. The ill-conditioning is a major reason of slow and inaccurate results from the back propagation algorithms for large data sets [34].

In a hybrid system of neural networks and rough sets, we use the benefit of rough sets to solve the problem about weights poorly initialized in neural network. By using reduct and core from rough set theory, we can decide which attributes are more important. Then, we can use the most important attribute to initialize the network weights. Moreover, rough set method can be used to extract rules from data. These rules can guide the network training, so that the system can reduce the number of iterations.

To sum up, it is hard to choose the way to build the hybrid system. Because the project

involves software in a cell phone, we need to think about the user experience. A well-designed hybrid engine is necessary for speedy convergence. Running the engine to train the neural network is also a critical problem.

3.2 User Interface

How to design the user interface is an important problem. We need to make the UI simple and clear. Figure 3.1 illustrates the draft design of UI:

Figure 3.1: Draft design of UI

There are three main UI in Figure 3.1. From left to right, they are song list, music player and developer UI. The song list UI and music player UI are for the user. The developer UI is for testing. However, how to make UI more friendly is very important and also we need to display all the information the user needs in one screen.

3.3 Portability

In this project, we need to implement an Artificial Neural Network (ANN) engine and a Rough Set (RS) engine. But the portability of these engines is very important. When we use the term ‘portability’ in this thesis, we mean software portability. Software portability is means that the engines are not only useful in this project, but also they are easy to be applied to other projects that need these engines like the projects in [3][4]. When some projects or software need an engine to analysis the data by using RS hybrid NN, they can use our engine to work on that because the hybrid engine is separated from the music player. It means that our hybrid engine cannot only work on the music data in our project, but also is able to process other data not related to music. In fact, our engine can be used on any tabular data in which we wish to learn patterns. In order to achieve that, the design of structure of code is very important. The software needs a good structure, so that the engine can be portable.

The ideal portable hybrid engine allows input of any data into this engine. We will generate some application-programming interface (API) for other developers. They only need to write a code about how to read their own data into the system, and then they can use our engine to analyze and generate a NN to solve their problem. To achieve this goal, we have carefully designed the structure of the code of the hybrid engine.

3.4 Database for user data and song data

In the project, a database was required to record user's operations and information. The design of the database became a very important issue. If the structure of the database is not well designed, there will be much work to do when we need to change the attributes of the database. If the database is not well designed, for example, lack of an important attribute, we may need much time to add a new attribute into the system once the database has been loaded.

Much information was needed to be recorded when a user is using this APP. For example, how many times the user skips the specific song, how many times the user plays a specific song, how many times the user has picked the song to be played, how many times the song has been play continually and so on. This information needs to be well organized in the system. When the NN hybrid RS engine needs to analyze data, it needs to be presented with the information combined and summarized in a table.

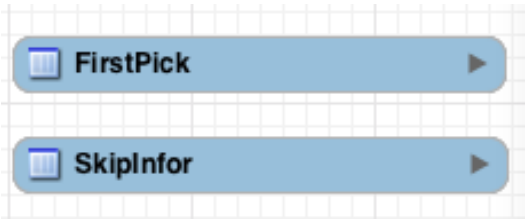


Figure 3.2: Draft design of database to record information about user's operations

Referring to Figure 3.2, firstpick table stores user's operation of picking the song on purpose, and skipinfor table stores every skip operation done by the user. These two tables are important parts of the system, because they record the valuable information

from user's operations with respect to a specific song.

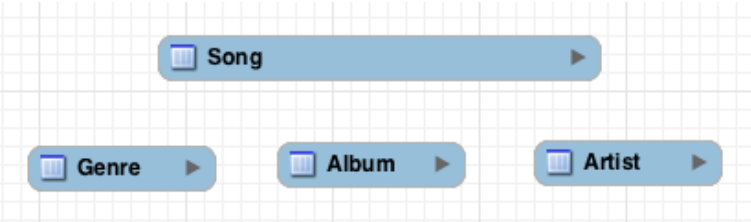


Figure 3.3: Draft design of database to record information about songs

Design of the database to collect song data is illustrated as Figure 3.3. The song table stores information about songs. The genre, album and artist tables as their names suggest store the genre, album and artist of songs.

3.5 Generation of playlist data

When we implement the NN with RS engine, we need some data to test the engines' performance. We needed to simulate some data. The data should simulate how people listen to music. The simulation process needs to follow some rules that express how a typical human being chooses and listens to music.

The data are partitioned into training and testing set. Then ANN with RS engine can be tested to see whether they can find the patterns that were intentionally introduced to the data.

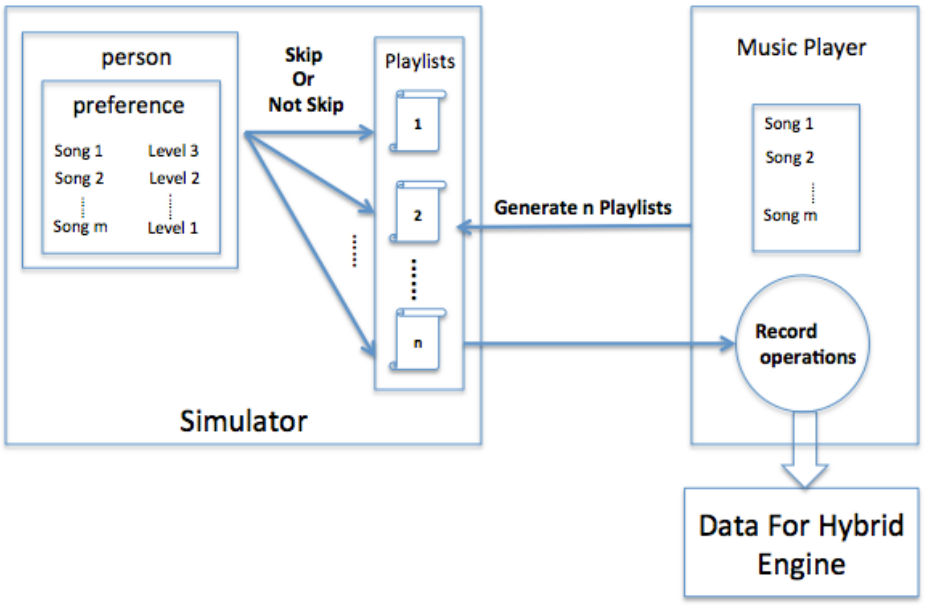


Figure 3.4: Diagram to describe simulation process

However, the process of generating data is not an easy job because it needs to simulate the listening process of people. Referring to Figure 3.4, we assume that a person will judge a song based on the degree to which this song is his favorite. A song is assigned a level to indicate degree of favor. For example, if the song is level 5 for this person, the song will hardly be skipped in the playlist. In contrast, if the song is level 1, the song will almost always be skipped in the playlist. Referring to Figure 3.4, the music player provides n playlists for the simulator. The simulator goes though n playlists, and simulates the skip operation based on the virtual person's preference of music. At the same time, the music player will record these operations as input data for the hybrid engine. Based on the assumption that level of a song for a particular user can be deduced from frequency of the user playing the song, we simulated a corpus of data for training and testing the NN hybrid RS engine.

To sum up, there were many challenges in developing this project. We needed to consider many issues when we were implementing our application. The hybridization is the most important part. The design of classes in the hybrid engine was very important. We need the engine to be portable. The second important problem is to create a simulation method to generate data for testing the engine. The solutions of these two problems are shown in the next chapter.

3.6 Shuffle Algorithm

There are two different shuffle algorithms in this project used for different purposes. The regular shuffle algorithm introduced in a former chapter is used to generate playlists. Referring to Figure 3.4, there are many songs that need to be shuffled to simulate different playlists.

But as a bonus, we want a shuffle algorithm that can be supervised for increasing the user's experience. In this project, the playlists so generated will not be totally random. By using the NN engine, we can get the song level to be predicted. The song level is a measurement for deciding how much the user likes this song. For example, level 5 means this song is the user's favorite. By using these predictions, we can supervise the special shuffle algorithm to generate a song list for the user that can provide the user with a better experience when using the music player.

To sum up, the project will provide two different ways to generate the playlist. One is the

normal shuffle algorithm, and it mainly is used in simulation process. The other one is for providing a bonus function to increase user's experience. When users are using the APP, they can change the mode to generate the playlist and enjoy it. This special shuffle algorithm is not like previous shuffle algorithms describe in previous Section 2.2. Original shuffle algorithm is to randomly arrange the sequence of song list. The new one will incorporate the song level to be a factor to generate the song list. It can be called shuffle under supervision. More information will be covered in Section 4.4.

Chapter 4

4 Problem Solution

In this section, we will present the implement of the project. In the overall design, we will briefly introduce the framework and design of the project. Then, we start from database structure and go through the implement of hybrid engine, supervised shuffle and the future work in detail.

4.1 Project design

4.1.1 Overall Design

Application software (APP) consists of a music player that is capable of self-learning and self-improving. This subsection describes the design of the application software.

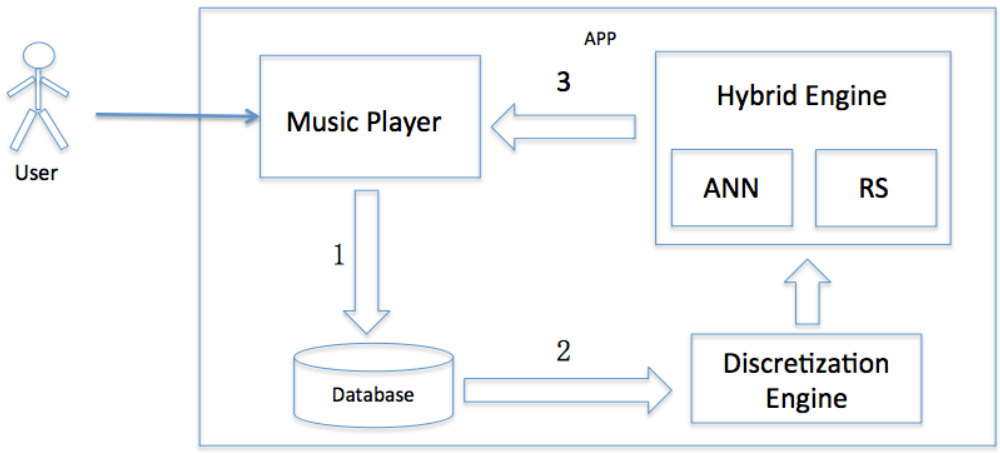


Figure 4.1: Overall design of self-learning audio player

Figure 4.1 uses a graph to describe how the whole system works. When people are using the music player, the system will record the operations people do like play song, skip song and so on. The system records these to the database. The data of the database are discretized and input into the hybrid engine. The hybrid engine will make the prediction of which songs are the user's favorite. The prediction is to assign the level to each song from 1 to 5 with 1 being the least favored and 5 being the most favored. These results will go back to the music player, so that it can perform the DJ mode shuffle also called supervised shuffle as we describe in Section 4.4.

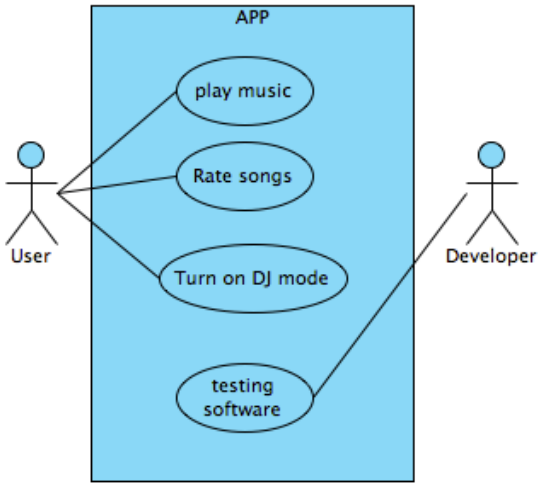


Figure 4.2: Use case diagram of system

The use case diagram of this project is illustrated as in Figure 4.2. Users can use the software, but only the developer can use the testing software function.

4.1.2 Application framework

Objective C programming language is an object-oriented programming language. When

we develop software in Objective C, we normally use a paradigm called MVC that is short for model, view and controller. Model represents the real object in real life, like song, artist, and album in our project. View is for presenting the interface to the user. Controller does all the communication jobs in the system.

Model-view-controller (MVC) has been widely used as a software development framework in different programming projects, for example, World Wide Web, Android and IOS applications. Applications of this framework vary in the way responsibilities are divided between the client and server. As illustrated in Figure 4.3, the controller receives a user's request. If the request is for data, the controller requests data based on what the user requests. In this way, the user is separated from the data itself making the software developed in MVC framework portable across databases.

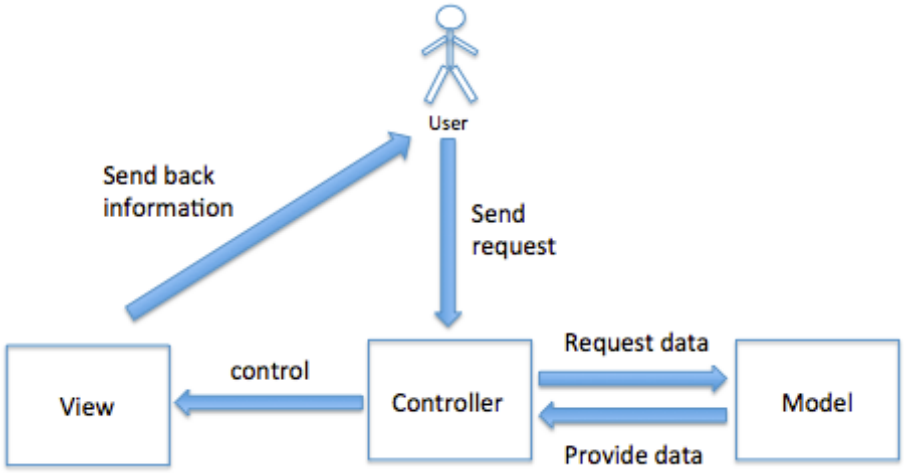


Figure 4.3: Diagram of MVC

Model provides data to the controller. Controller will process the data according to the user's demand, and then control the view by which to display the result. In the end, the

view will show the user the result. Differ form MVP (Model – View-Present), the controller in MVC is responsible for determining which view response to any actions. In MVP, the presenter contains the UI business logic for view, so that the controller in MVP only does mediate job between view and controller.

4.1.3 Class Design

In this subsection, we describe the design of important classes such as Models, Music Player Controller and Database Manager. Models represent objects being modeled such as song, artist, album and genre.

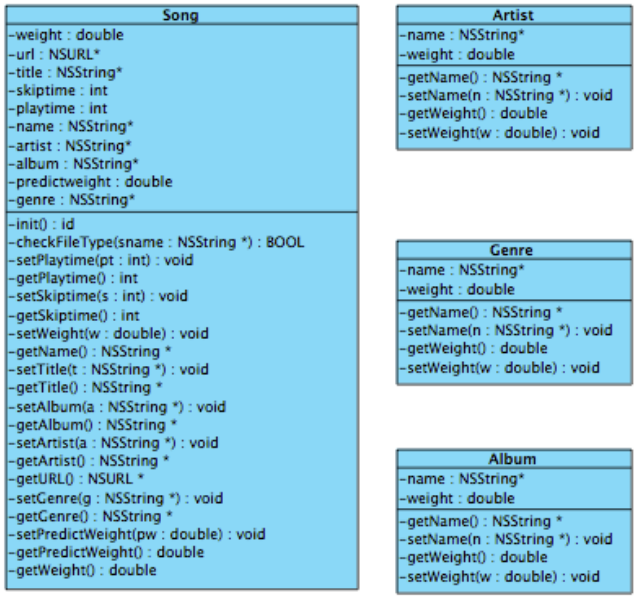


Figure 4.4: Class diagram for Models

Referring to Figure 4.4, there are 4 classes. They are all models that represent real world objects such as song, artist, genre and album. When the system is running, they will be used to create instances of objects. For example, there are 50 Song objects in the system

when the number of songs in the music player is 50. Each instance of Song class has information about the exact song it represents. So, when the controller needs data, they can come directly to these classes to get the data.

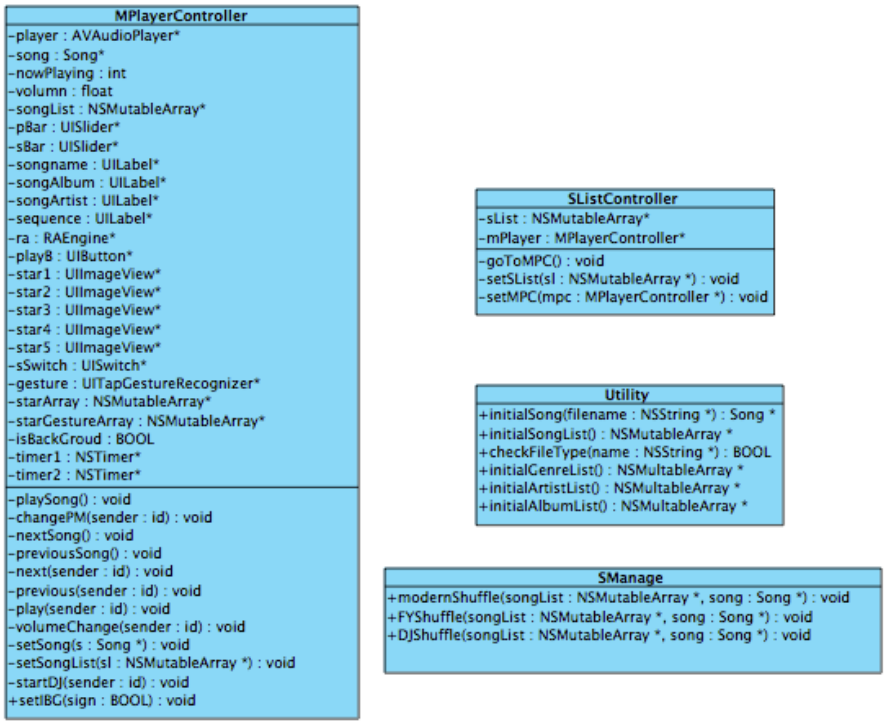


Figure 4.5: Class diagram for the Controllers and helper classes

Referring to Figure 4.5 there are two main controllers of the system and two helper classes. Helper classes are reusable and help us to develop the software. There are some functions that are very useful for more than one situation. We make a helper class for these functions, so that every other class that needs them can use them. The description of each class is given in Table 4.1.

As was illustrated by Table 4.1 illustrated, MPlayerController is the main controller. When a user wants to play the next song, play the previous song or stop playing the song,

MPlayerController execute these operations. Moreover, MPlayerController will display the name, album, artist and genre information of the song to the user. There are a lot of other private functions of this controller, like activate hybrid engine and DJ shuffle mode. SListController shows the user the song list. The user picks a song, and then SListController will send the user's choice to MPlayerController to play the song.

Table 4.1: Description of controller and helper classes

Class	Description
MPlayerController	The most important controller, it is used to process all the operations of the music player
SListController	Controller for the song list, it helps to list songs and process the operations when people pick a song.
Utility	A helper class that provides initialization functions.
SManage	Allows developer to choose between normal shuffle and DJ mode.

Figure 4.6 is presenting the structure of the database manager. We created a parent class that implements some basic operations like open database and close database. The other classes inherit the functions of the parent class.

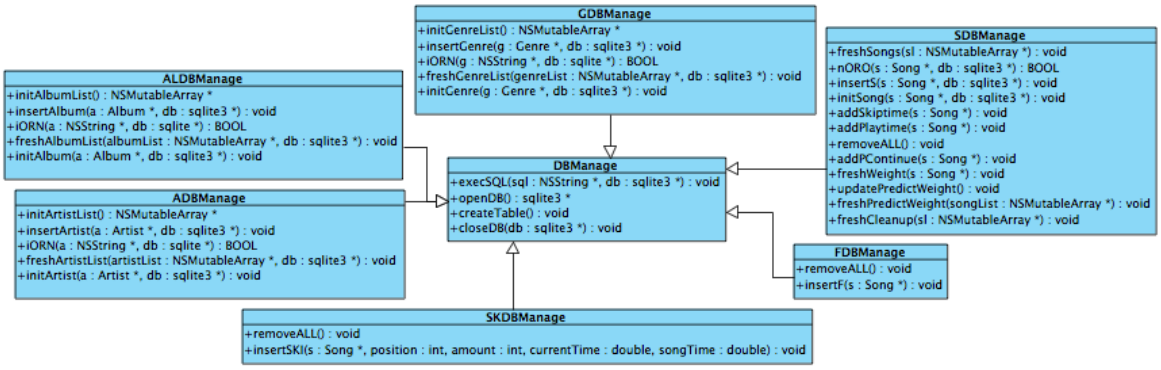


Figure 4.6: Class diagram of Database manager

Each child class is associated with a table in the database, for example, Song table is

associated with SDBManage, Artist table is associated with ADBManage.

4.1.4 Design of the User Interface

When the user opens the APP, the main UI is the song list. All the songs in user's library will be listed. The Song List UI is as illustrated in Figure 4.7.

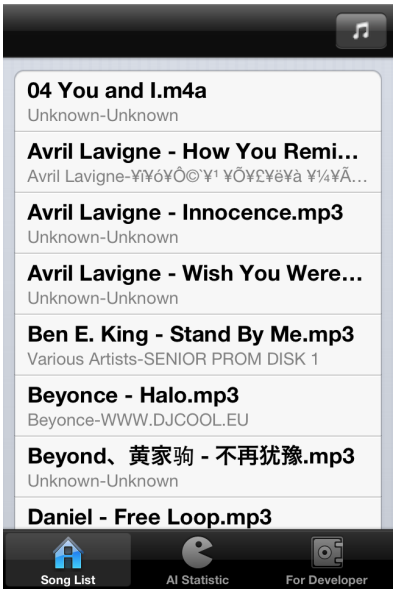


Figure 4.7: Song List UI

There are three tabs along the bottom, which are Song List, AI statistics and an evaluation tool for the developer. When the user clicks the elements of the list, APP will go to the Music Player UI to play the song that the user has chosen. Also, the user can click the button with the music sign in top right corner to go to the Music Player UI. The Music Player UI is as illustrated in Figure 4.8:

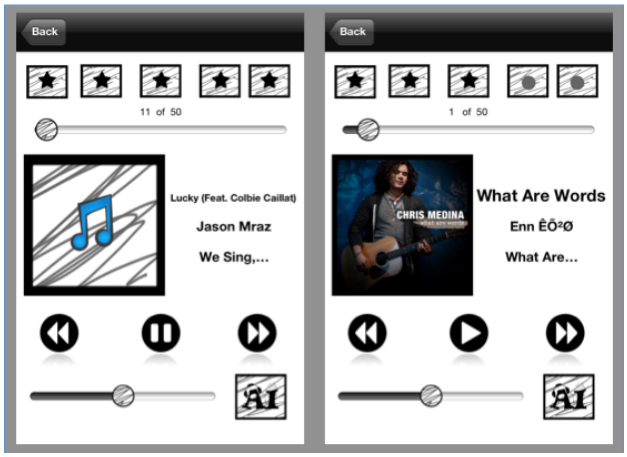


Figure 4.8: Music Player UI

Referring to Figure 4.8, the user can rate any songs by touching the number of stars to give its level. In the left picture of Figure 4.8, the level of the song is 5, and the right one is level 3. The slider at the top is to show the duration of the song. The user can slide it to choose which part of the song to play. The other slider in the bottom is to change the volume of the music. When the Audio file contains the cover picture (not all do), that picture will be displayed in the left as shown. The right part displays the meta data, which include title, artist and album information of the song.

When the user uses the music player to listen to music, the APP will record the operations the user does. The sequence diagram of Figure 4.9 shows how the system works when the user picks a song and selects the operations such as play the next song or play the previous song.

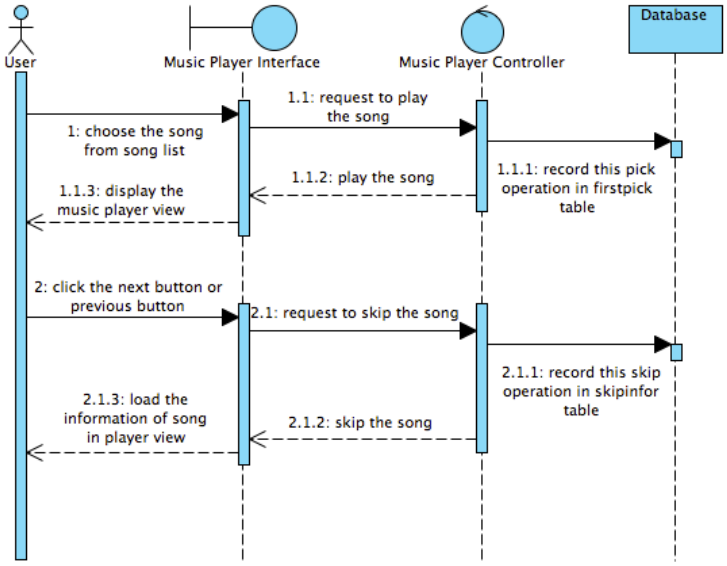


Figure 4.9: Sequence diagram of basic operation of Music Player

In the bottom right corner of the Music Player UI, we can see a square picture with AI labels. When the user touches it, a switch appears to allow the user to open the DJ mode shuffle, which will be introduced in Section 4.4.

Figure 4.10: The switch of DJ mode

Referring to Figure 4.10 the left picture shows the UI before the user tries to open the DJ mode. The right picture shows the UI after the user touches the AI graph and the switch is showing on the user interface in the right picture.

When the APP goes to the background, the timer will activate the hybrid engine. The hybrid engine will start to analyze the data. The reason to activate the timer after APP goes to background is that the hybrid engine will occupy a large amount of memory and it is computationally intense. When the application is running in background, the user can still listen to music as long as he does not touch the screen. The resources dedicated to the user interface are free to use in the background application. Otherwise, the application will influence user's experience.

The sequence diagram of Figure 4.11 shows the process of self-learning:

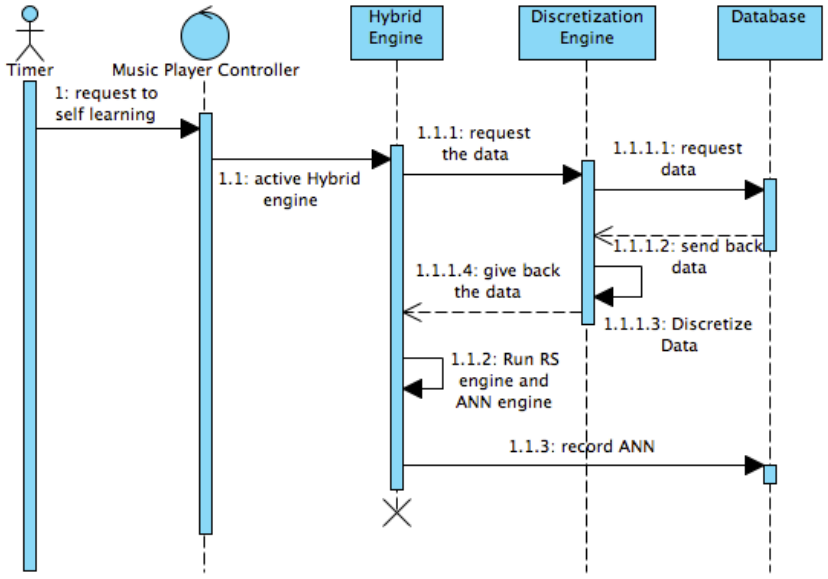


Figure 4.11: Sequence diagram for self-learning

Referring to Figure 4.11, before we use the hybrid engine to analyze the data, we perform a discretization method as we described in Section 2.5. The discretization engine will request data from the database and create a table. The table in Figure 4.12 is an example of data before discretization. The information in this table is a summary of information of

song, skipinfor and firstpick tables, and it is suitable for input to ANN. The columns of Figure 4.12 were described in Table 4.6 in Chapter 4.

songname	rateartist	ratealbum	rategenre	skiptime	playtime	pcontinue	scontinue	maxpcontinue	maxscontinue	firstpicktimes	ratesp	userrate	desirerate	predictrate
04 You and I.m4a	0.42	0.52	0.64	17	50	0	1	9	2	1	0.34	4	0	0
Avril Lavigne - How You Remind Me.mp3	0.02	0.02	0.64	16	51	1	0	6	2	1	0.3137254901...	4	0	0
Avril Lavigne - Innocence.mp3	0.42	0.52	0.06	45	52	1	0	2	12	1	0.8653846153...	1	0	0
Avril Lavigne - Wish You Were Here.mp3	0.42	0.52	0.64	37	51	1	0	2	9	2	0.7254901960...	2	0	0
Ben E. King - Stand By Me.mp3	0.02	0.02	0.64	18	51	5	0	12	5	4	0.3529411764...	4	0	0
Beyonce - Halo.mp3	0.02	0.02	0.02	3	50	26	0	17	1	0	0.06	5	0	0
Beyond、黄家驹 - 不再犹豫.mp3	0.42	0.52	0.64	45	52	0	1	1	13	3	0.8653846153...	1	0	0
Daniel - Free Loop.mp3	0.42	0.52	0.64	24	51	1	0	4	4	1	0.4705882352...	3	0	0
Green Day - The Forgotten.mp3	0.42	0.52	0.64	8	51	1	0	18	0	3	0.1568627450...	5	0	0
Jack Johnson - Better Together.mp3	0.02	0.02	0.02	39	50	1	0	4	12	0	0.78	1	0	0
Jessie J - Price Tag.mp3	0.02	0.02	0.64	25	50	0	3	4	4	1	0.5	3	0	0
Jessie J - We Found Love (Rihanna Cover).mp3	0.42	0.52	0.64	5	50	7	0	12	0	2	0.1	5	0	0
Louis Armstrong - What A Wonderful Wor...	0.02	0.52	0.04	28	50	1	0	4	6	2	0.5600000000...	3	0	0
Lucky - Jason Mraz&colbie Caillat.mp3	0.02	0.02	0.18	19	50	0	1	4	2	2	0.38	4	0	0
Maddi Jane - Price Tag.mp3	0.02	0.52	0.18	43	50	0	15	2	18	0	0.86	1	0	0
Natasha Thomas - It's Over Now - 鲸鱼广...	0.04	0.02	0.02	35	51	1	0	5	12	2	0.6862745098...	2	0	0
Natasha Thomas - Let Me Show You The	0.42	0.52	0.64	10	51	3	0	11	2	1	0.1960784313	5	0	0

Figure 4.12: Table to be discretized before input to ANN

Before we put the table of Figure 4.12 into ANN, we perform a discretization method named equal frequency binning discretization method that can be checked in Section 2.5.

songindex	songname	rateartist	ratealbum	rategenre	skiptime	playtime	pcontinue	scontinue	maxpcontinue	maxscontinue	firstpicktimes	ratesp	desirerate	userrate
1	01 Fever.m4a	3	3	4	2	3	4	5	4	1	5	2	2	2
2	01 The Best Is...	3	3	4	2	5	2	5	4	3	3	2	2	2
3	03 Georgia On...	3	3	4	5	1	5	1	1	5	1	5	5	5
4	03 Me and Mrs....	3	3	4	5	1	2	5	1	5	1	5	5	5
5	Ben E. King - St...	5	5	4	4	1	5	2	1	5	1	4	4.5	5
6	Beyonce - Halo....	5	5	5	1	1	5	1	4	1	2	1	1	1
7	Jessie J - Price...	5	5	4	3	1	2	5	4	4	1	3	3	3
8	Jessie J - We Fo...	3	3	4	3	5	5	2	4	1	5	3	2.5	2
9	Louis Armstron...	5	3	5	5	3	3	5	1	5	1	5	5	5
10	Lucky - Jason...	5	5	5	4	3	2	5	2	4	3	4	4.5	5
11	Maddi Jane - Pr...	5	3	5	1	5	5	1	5	2	5	1	1.5	2
12	Sam Tsui、201...	5	3	4	3	5	3	5	2	3	4	3	3.5	4
13	The Voice - If I...	5	5	5	5	3	1	5	1	5	3	5	5	5
14	What are words...	5	5	5	2	5	4	5	5	2	2	2	1.5	1
15	世界各地街头艺...	3	3	4	5	3	1	5	2	5	4	5	5	5
16	周杰伦 - 乌克丽...	3	3	4	1	5	3	5	5	1	5	1	1	1
17	周杰伦 - 傻笑.mp3	5	5	4	3	3	3	5	3	2	3	3	3	3

Figure 4.13: Discretized Table before input to ANN

The table of Figure 4.13 will be put into the ANN for training. All the data in this table have been discretized. After discretization, the data will be sent back to the hybrid engine, and then the hybrid engine runs the self-learning function and records the ANN to the database for the prediction.

There are some more sequence diagrams that can be found in the Appendix 2. They are good to help to understand how this APP works in some specific situations.

4.2 Database

In this project, we use SQLITE database. SQLITE is a software library. It can provide functions for people to use a SQL database engine. SQLITE is the most widely deployed SQL database engine in the world.[35] According to what we need to achieve in this project, our database has been designed as illustrated in Figure 4.14, 4.15, 4.16 and 4.17.

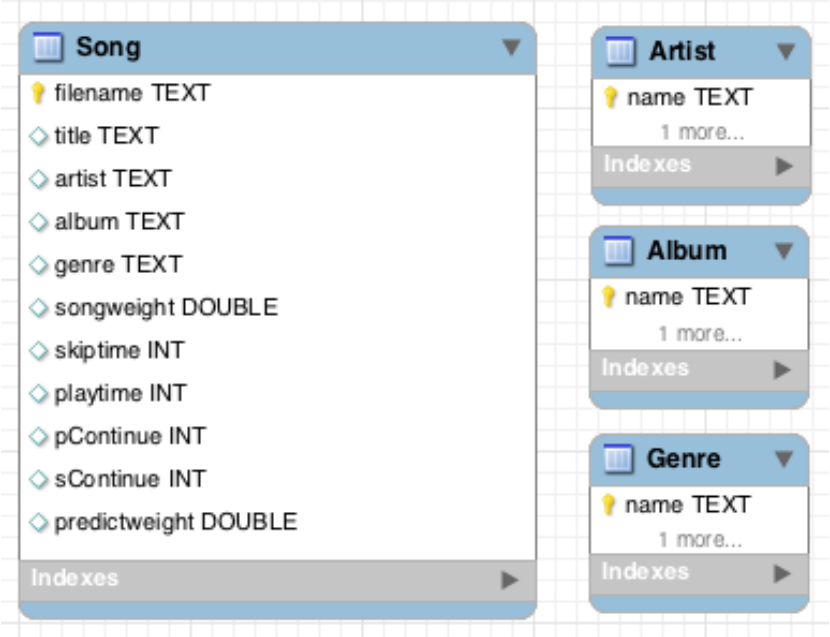


Figure 4.14: Detail design of database (A)

Referring to Figure 4.14, we have presented a detailed design of Song, Artist, Album and Genre tables. In Artist, Album and Genre tables, the meanings of attributes can be deduced. In Artist, Album and Genre tables, we store the name of artist, album and genre of songs. Redundancy is avoided by using this design because, for example, if an album is named “white house” the name “white house” will appear only once in the Album table. The limitation in this design is that if the two different artists have the same name in their

album, the system cannot distinguish them. Artist and Genre tables are the same. By using these three tables, we can make statistics about how many songs have the same genre, or how many songs are in the same album or how many songs belong to the same artist. Song table has many attributes, and they are all very valuable. The attributes for Song table are illustrated as Table 4.2.

Table 4.2 Description of Song table

Attributes	**Description**
Filename	The name of the audio file in which the music is stored
Title	The title of song, it can be extracted from song's audio file.
Artist	The artist of song, it can be extracted from song's audio file.
Album	The album of song, it can be extracted from song's audio file.
Genre	The genre of song, it can be extracted from song's audio file.
Songweight	The level of song indicating degree of favor, it is given by user
Skiptime	How many times has this song been skipped.
Playtime	How many times has this song been played.
Pcontinue	How many times has this song played continually recently.
Scontinue	How many times has this song been skipped continually recently.
Predictweight	The prediction level by ANN/RS engine.

As Table 4.2 shows, the attributes of the song table provide basic song information, but more importantly provide information that can be used by the system to predict a song's level. When the system is running, song table will provide the data input to the hybrid engine.

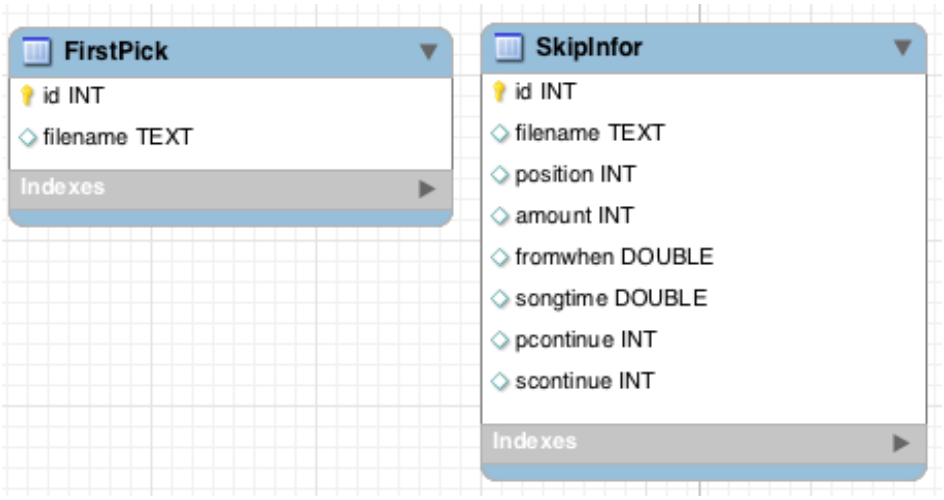


Figure 4.15: Detail design of database of Firstpick and Skipinfor tables

Table Firstpick and Skipinfo are as illustrated as Figure 4.15. They store every operation from user about skipping and picking songs. Tables 4.3 and 4.4 provide the description of every attribute of these two tables.

Table 4.3: Description of Firstpick table

Attributes	**Description**
id	Artificially generated key for the first picked songs in a playlist
Filename	The name of the audio file in which the music is stored

Table 4.4: Description of Skipinfor table

Attributes	**Description**
id	Artificially generated key for whenever the user skips the song
Filename	The name of the audio file in which the music is stored
Position	When the song is skipped, in which position is it in the playlist.
Amount	How many songs were in this playlist.
Pcontinue	How many times this song has not been skipped previous to it being skipped this time.
Scontinue	How many times has this song been skipped previous to when it has been skipped this time.

In Table 4.3, the descriptions are given of attributes that record a user's operations of

picking a song. Because people tend to pick their favorite song first to listen to, the record in the first pick table is a valuable resource. The frequency of appearance of songs is important, because it indicates how much the person likes the song.

In Table 4.4, the fields pcontinue for play continuously, and scontinue for skip continuously will be described using an example. For pcontinue, if the user plays the song continually to the end 4 times, and then skips it, the pcontinue value will be set from 4 back to 0. For scontinue, if the user skips the song continually 10 times, and then plays it to the end once, the scontinue value will be set from 10 back to 0. These 2 attributes are important, because they show that whether this song is favor for the user. For example, if the user keeps skipping one song every time, the user may feel bore for this song already. In contrast, if the user never skips one song, the user at least does not feel bore for this song.

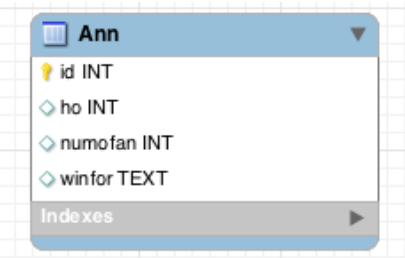


Figure 4.16: Detail design of database of ANN table

As illustrated in Figure 4.16, table ANN is used to record the weights and thresholds of every neuron in the neural network. After training the ANN, the system records the ANN into this table. When we want to use the ANN to predict the other data, this table is used to initialize the neural network. The descriptions of attributes of ANN are shown in Table

4.5.

Table 4.5: Description of Ann table

Attributes	Description
Id	Artificially generated key
Ho	A sign to separate whether the neuron is in output layer or in hidden layer
Numofan	The position of the neuron in its layer
Winfor	The value of weight and threshold of the neuron.

In Table 4.5, the attribute Ho is a sign to know the layer the neuron is in. For example, if the neuron is in the hidden layer, its Ho will be 0. If the neuron is in the output layer, its Ho will be 1. Though we have an input layer, in our application, there are no neurons in it. The attribute Numofan records the position of the neuron. For example, if the neuron is the third neuron in the hidden layer, the Numofan will be 2. If the neuron is the first neuron in the hidden layer, the Numofan will be 0. The attribute Winfor records the values of a neuron's weights and its threshold. The values are stored in the following format:

$$1.785981\sim2.368342\sim1.305402\sim-1.236647\sim-2.355511\sim2.183669\sim0.113737$$

In the above example there are 7 values and they are divided by character '~'. This is a neuron in the output layer. The neuron has 6 weights for its input, and one threshold. So, the first 6 values are the values of the weights, the seventh one is the value of the threshold.

Referring to Figure 4.17, this is the detailed design of dsong table. The table provides a summary of information of song, skipinfor and firstpick tables, and it is suitable for input

to NN. Some of the attributes values come directly from the three tables; some come from a summary of information in the three tables.

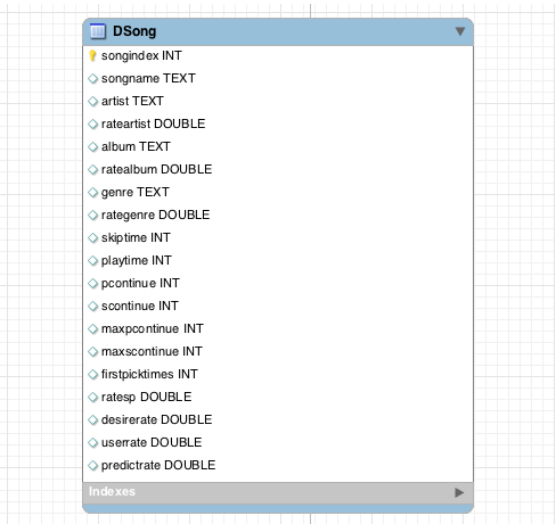


Figure 4.17: Detail design of database (D)

The Table 4.6 is a table showing description of each attribute in dsong. It will reveal how each attribute is materialized. The dsong table is defined from the other tables in the database.

The attribute userrate is how many levels the user rates this song. If the song has not been rated, it will be 0. The level is from 1 to 5. The ratesp is the ratio of skiptime and playtime. The relation between skiptime and playtime is as follows:

$$skiptime + (how\ many\ times\ the\ song\ has\ been\ played\ till\ the\ end) = playtime$$

So, ratesp provides a very simple knowledge to show whether this song is favored. When the user does not rate the song, the desirerate will be ratesp. If user rates the song, the desirerate will be as Table 4.6 declares. The attribute desirerate is used as the decision value in ANN hybrid rough set engine.

Table 4.6: Description of Dsong table

Attributes	Description
songindex	Artificially generated key for each song
Songname	The name of the audio file in which the music is stored
Rateartist	The ratio of this song's artist. $\text{rateartist} = \frac{the\ number\ of\ this\ artist's\ songs}{the\ amount\ of\ songs}$
Ratealbum	The ratio of this song's album. $\text{Ratealbum} = \frac{the\ number\ of\ this\ album's\ songs}{the\ amount\ of\ songs}$
Rategenre	The ratio of this song's genre. $\text{Rategenre} = \frac{the\ number\ of\ this\ genre's\ songs}{the\ amount\ of\ songs}$
Skiptime	How many times has this song been skipped? The value comes directly from the song table.
Playtime	How many times has this song been played? The value comes directly from the song table.
Pcontinue	How many times has this song been played continually recently? The value comes directly from the song table.
Scontinue	How many times has this song been skipped continually recently? The value comes directly from song table.
Maxpcontinue	The maximum times this song has been played continually. It comes from skipinfor table.
Maxscontinue	The maximum times this song has been skipped continually. It comes from skipinfor table.
Firstpicktimes	How many times has this song has been first picked? It comes from firstpick table.
Ratesp	The ratio of skiptime and playtime. $\text{Ratesp} = \frac{skiptime}{playtime}$
Desirerate	This is what the level of the song should be. This attribute is mainly used in training ANN but not in testing. $desirerate = \frac{(ratesp + userrate)}{2}\ (if\ userrate > 0)$ $desirerate = ratesp\ (if\ userrate = 0)$
Userrate	This is what user rates the level of song to be.
Predicterate	This is what ANN predict the weight of song to be.

4.3 The Artificial Neural Network hybrid Rough Set Engine

4.3.1 Artificial Neural Network Engine

The algorithm of artificial neural network (ANN) has been shown in Chapter 2. In the project, we will use a multilayer neural network. In order to make ANN engine portable, we designed the engine as illustrated in Figure 4.18.

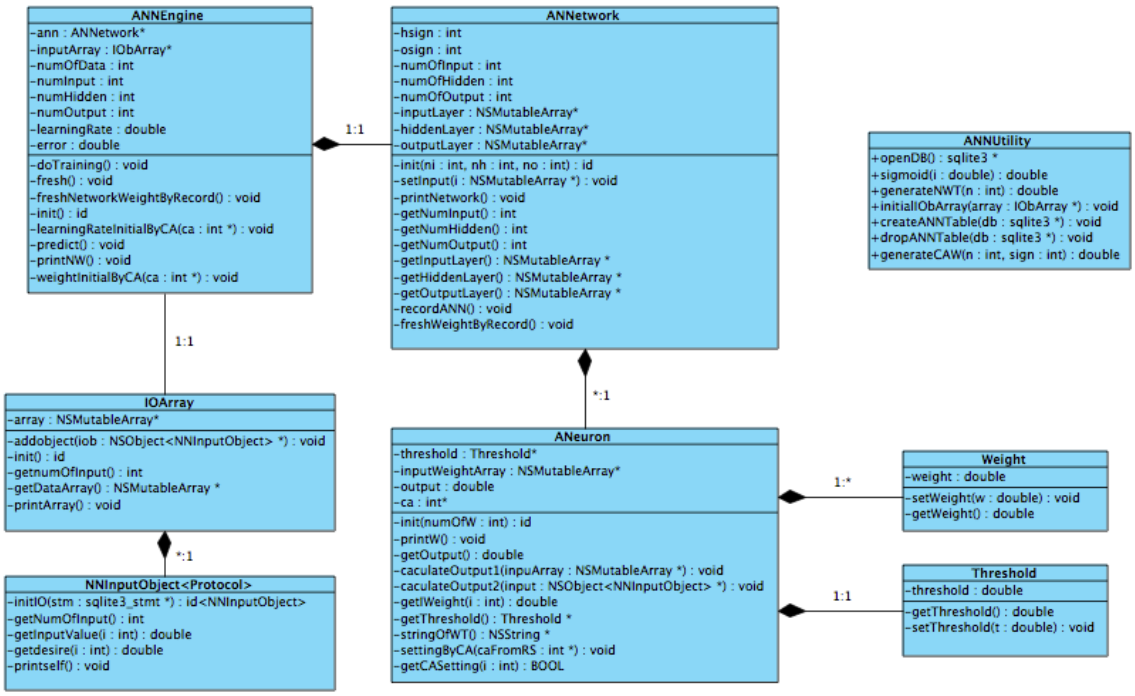


Figure 4.18: The class diagram of ANN Engine

As illustrate in Figure 4.18, there are 8 classes in this engine. They all have their own meanings in this ANN Engine. The description of each class is as given in Table 4.7. For a portable design, if other software wants to use the hybrid engine, it must define a class to implement the protocol exactly as seen in lower left of Figure 4.18. When the class implements the protocol, the class has to implement the methods that the protocol

describes. This is a strategy of my design to make the code portable.

A neuron has multiple weigh, a weigh can be seeing as a multivalued attribute. The weight of a neuron is implemented as an object not an attribute, because then it is easy to change the structure of the NN landing greater portability to the hybrid engine.

Table 4.7: Description of classes in ANN

Class	**Description**
Weight	It represents the weights in the neuron; one neuron has several weights.
Threshold	It represents the threshold in the neuron; one neuron has only one threshold attribute.
ANeuron	It represents the neuron in the neural network; one neural network has several neurons.
ANNetwork	It represents the neural network. In ANN engine, only one neural network exists.
NNInputObject <Protocol>	It is a protocol. Every class that is implemented in this protocol implements all the operations it mentions. It is a protocol for the input object.
IOArray	It is an array that store input objects.
ANNEngine	It represents the ANN Engine. The self-learning, prediction operations are implemented in this class.
ANNUtility	It is factory class that provides some commonly used functions for the other class.

As we mentioned in Chapter 3, the portability of ANN engine is very important. Referring to Figure 4.18 the other people who want to use this engine only need to do a little programming, and then the engine will work for their data. The programmer needs to do the following steps:

Step 1: Code their own class to implement the NNInputObject<Protocol>.

Step 2: Recode the initialIObArray and openDB function in ANNUtility class.

Step 3: Recode the init function in ANNEngine to define how many neurons are in the hidden layer and/or in the output layer, the learning rate and error that is allowed to exist.

Step 4: Use the functions in ANNEngine, they are already able to use the engine to do what they want to with their data.

4.3.2 Rough Set Engine

The algorithm of rough set inductive learning was shown in Chapter 2. As we declared in Chapter 3, we need to make RS Engine to be portable which we have done as illustrated in Figure 4.19

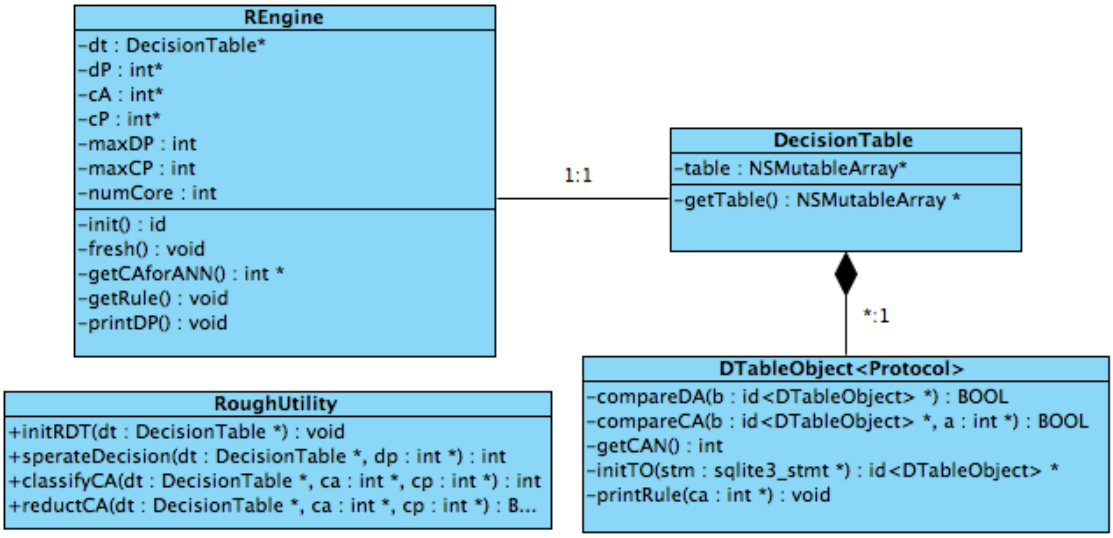


Figure 4.19: The class diagram of RS Engine

There are 4 classes in this engine. The description of each class is as Table 4.8 shows.

Table 4.8: Description of classes in RS

Class	Description
DTableObject <Protocol>	It is a protocol. Every class that is implemented in this protocol implements all the operations it mentions. It represents each object that is input to RS.
DecisionTable	It represents the array that stored DTableObject. One DecisionTable has several DTableObject.
REngine	It is the engine to run rough set engine. One rough set engine has one DecisionTable. It also performs the getRule function
RoughUtility	It is a factory class that provides some common functions for the other classes.

By ”core the decision table”, we mean to reduce size of decision table to only those attribute that are meaningful for deciding the level of the song. The rest are redundant providing no additional information of how much the user likes the song. By using the Rough Set Engine, we core the decision table and give a set from getCAforANN function to REngine. The set will be like:

$$ca[6] = \{0,1,1,1,0,1\}$$

The list of zeros and ones mean there a 6 conditional attributes. Besides the first and the fifth elements in the list both of which is 0, the attributes are all significant attributes indicated by 1 in the list. When we run hybrid RS Engine and ANN engine, we input this attribute to ANN to help it run. To be portable, the other users only need to do a little coding, and then they can use RS Engine. The steps are as follows:

Step 1: Code their own class to implement the DTableObject<Protocol>.

Step 2: Recode the initialRDT and openDB function in RoughUtility class.

Step 3: Using the functions in REngine, they can use this engine to analyze data, generate the rules, and find out the significant attributes.

4.3.3 The Hybrid between ANN and RS

The way to hybrid ANN and RS is that we use the ability of Rough Set to core the conditional attributes and find out the significant ones. Then, we use this to help ANN training. The configuration of NN is that there are 6 neurons in hidden layer and 1 neuron in output layer, and the learning rate is 0.1. According to Section 3.1, we know that one hidden layer is sufficient to deal with the complex problem. The more neurons in hidden layer, the more precise the NN will be. But, in the same time, the more neuron will cost more computer resource. Referring to [3], they found out 5 neurons in hidden layer give the better results in their problem. Because our NN need to process more input, we increase the number of neurons in hidden layer. We make it good enough to solve the problem, but also do not cost too much computer resource. The following is an example of how we use the conclusion researched by the rough set method. The whole process is illustrated as Figure 4.20

Step 1: We discretize the decision table by using equal frequency binning discretization method.

Step 2: We put the data into RS engine. Firstly, we core the conditional attributes, and then we generate the rules. Secondly, we input the conclusion of coring to ANN engine.

As we discussed before, we can get a set of attribute names from REngine by using getCAforANN function. That set helps to initialize the weights of ANN engine. We will initialize the weight as 0 when the attribute is insignificant.

Step 3: ANN engine uses the data from discretization and the input from RS engine to train itself. The result from RS engine will help ANN to initialize weights and also guide the training process. In the training process, we will assign the learning rate of insignificant attributes to be 0. It means that we will not do any weight corrections for insignificant attributes. So, from the beginning of the training process to the end, the insignificant attributes will not influence the whole system.

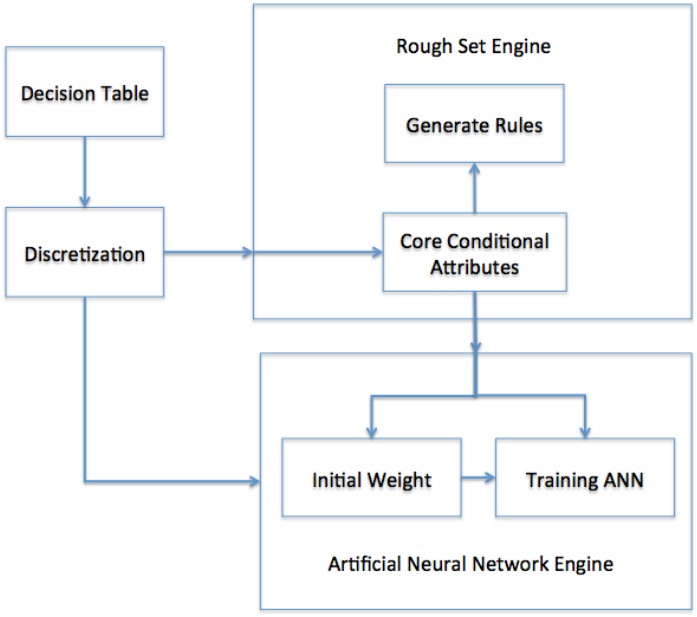


Figure 4.20: The process by which the hybrid engine runs

As said in Chapter 2 says that we will randomly initialize the values of weights. By using the result of Rough Set, the value of weights of insignificant conditional attributes will be

initialized as 0. Moreover, when we do the ANN training weight part, we will not do any change of the weight of these insignificant attributes. It means that we will not let the insignificant conditional attributes influent the ANN until Rough set says it becomes significant again.

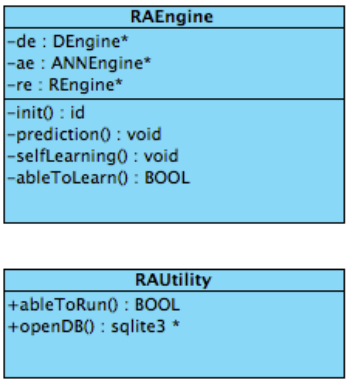


Figure 4.21: The class diagram of RAEngine

As in Figure 4.21 the RAEngine consists of 2 classes. RAEngine is a class that combines rough set engine and artificial neural network engine. A part of the code in RAEngine is illustrated as follows:

```
-(void)selfLearning
{
	[de discretization];
	int * ca;
	[re fresh];
	[re getRule];
	ca = [re getCAforANN];
	[ae fresh];
	[ae learningRateInitialByCA: ca];
	[ae freshNetworkWeightByRecord];
	[ae printNW];
	[ae doTraining];
	[ae printNW];
}
```

It is the main class of the hybrid engine. To be portable, the other users only need to recode the openDB function in RAUtility, and also they have to recode the REngine and ANNEngine like we mention below. Because we need to execute the discretization function before execution of RS process, so we do the discretization in RAEngine. For the other users, they should delete the first line. Then, they can use their own discretization method to preprocess their data.

4.4 The supervised shuffle

We designed a specific shuffle algorithm to shuffle the playlist. It is called supervised shuffle or DJ mode. This algorithm aims to improve the user experience by allowing the user to experience songs from different levels in a short section of songs rather than a whole library. The sections organize the songs from library. Initially, the algorithm separates songs into sections, so that there are songs in 1 to 5 different predicted levels. We make 5 songs as a section, and then we randomly put one song from each level in this set. It means that we randomly designate the indexes in a section. For example, the index in the section should be $[1, 2, 3, 4, 5]$. However, we might shuffle it to be $[3,4,1,2,5]$.

The supervised shuffle algorithm will use the predicted level of the songs output from the hybrid engine. The songs are separated into 5 groups (not to be confused with sections) based on their predicted level 1 to 5. Then, we randomly move a song in each group to the playlist. The detailed algorithm is as follows:

Supervised shuffle Algorithm

The songs are separated into 5 groups based on their predicted level 1 to 5
We assume 5 songs as a section, so $section = 5$
n: the number of songs that need to be shuffled.
$shuffle\ playlist$: output of this algorithm
$group[5]$: the set of groups
While (there are still songs to be placed)
Generate a random set of indexes (a permutation) from 1 to 5; //for example: $set = [1,4,5,3,2]$
For i **from** $section - 1$ **to** 0
$int\ j = set[i]$;
If $(group[j-1] \neq \emptyset)\{$
randomly move a song from $group[j-1]$ to $playlist$
continue;
}
Else{ //This group has no more elements
Flip a corn to decide whether to go higher or lower level
If (it is possible to move a song from a lower\higher level group then randomly move it)
//For example, if $j = 4$, and there is no element in group 4, we start
//randomly moving a song from group 3 and continue until we find a
//group less that j that is not empty.
Continue;
Else{
If (it is possible to move a song from a higher\lower level group then randomly move it)
Continue;
Else{
mark that all the songs have been placed and finish the algorithm;
}
}
}
End For
End While

We use an example to explain this algorithm clearly. We assume for illustration the coin always flips to go to the lower level first. In this example, we have 10 songs in the library.

So, we will separate the library into 2 sections. In these 10 songs, there are zero level 1 songs, two level 2 songs, four level 3 songs, three level 4 songs and 1 level 5 song. We split these songs in the 5 groups as group i stores level i songs. $(i = 1,2, \dots 5)$ The heading "Require for songs with level" means those songs that are required to be put in that level.

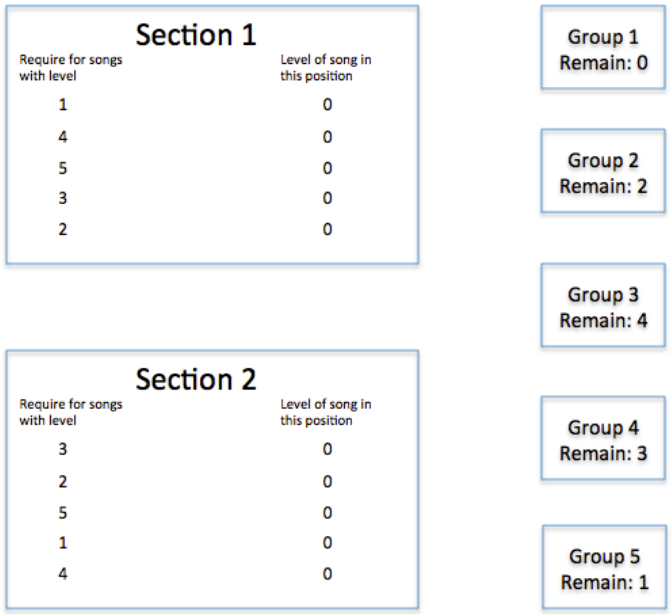


Figure 4.22:Generate random set of indexes for section

As can be seen in Figure 4.22, each section has 5 songs with different levels. The order of 5 songs is randomly generated. The order is shown on the left side of the section block. In section 1, the order is $[1,4,5,3,2]$.

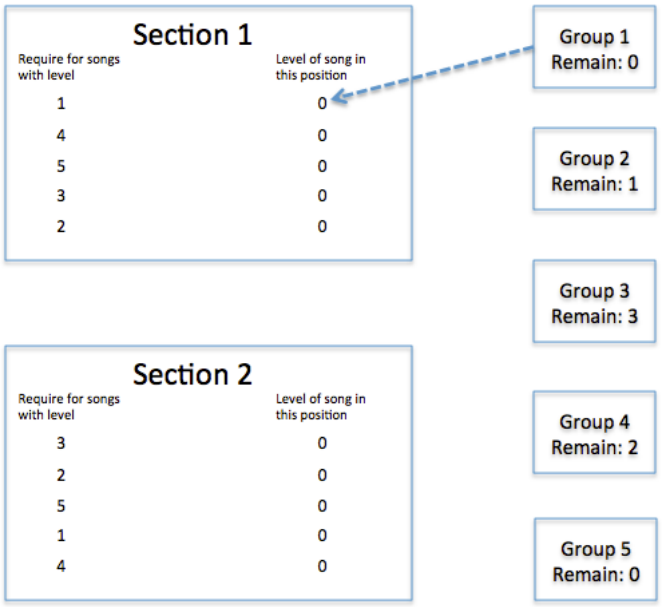


Figure 4.23: Moving songs from a group which is empty

In section 1 of Figure 4.23, the first song we want is a level 1 song. However, group 1 does not have any elements, and level 1 is the lowest level. So, we go to group 2 to randomly pick a song to put in the first song position in section 1.

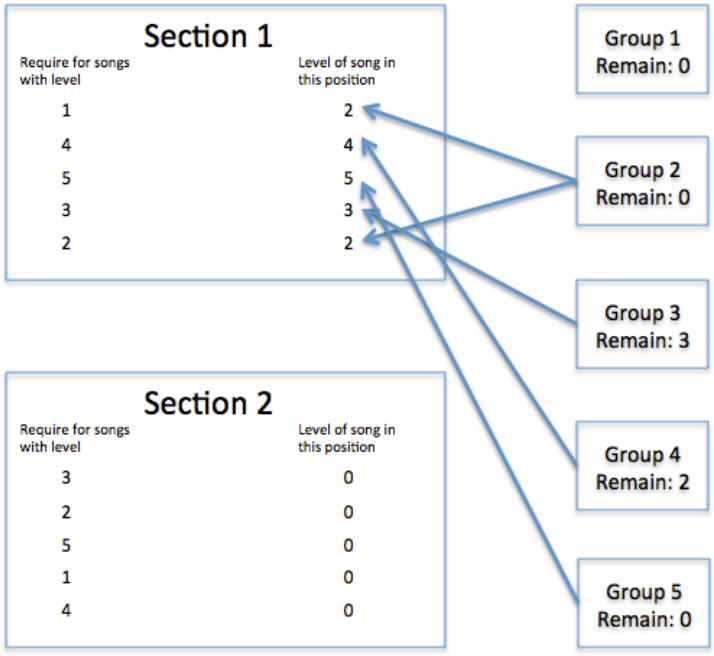


Figure 4.24: Moving songs with groups which is not empty

As can be seen in Figure 4.24, the songs in section 1 are level 2, 4, 5, 3, and 2. Because of that, only group 3 and group 4 have unassigned songs.

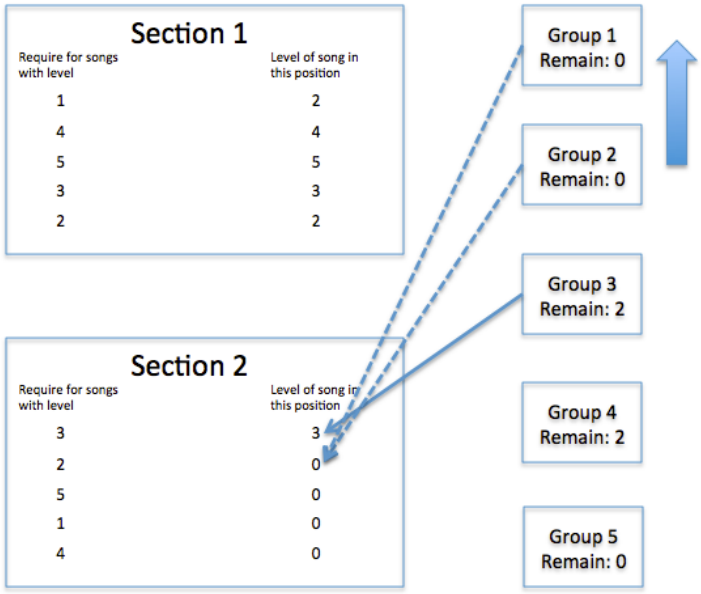


Figure 4.25:Moving a song from lower level group

As shown in Figure 4.25, we need to process section 2. The arrow shows the direction of movement of songs. The order of songs in section 2 is [3,2,5,1,4]. After we randomly pick a song from group 3 to put into the first position, the second position requires level 2 songs. Group 2 is empty, so the algorithm will go to the lower level group to find songs. However, Group 1 is also empty, and level 1 is the lowest level. Therefore, the algorithm will go to the greater level to find songs. As Figure 4.26 shows, the algorithm randomly picks a song from group 3.

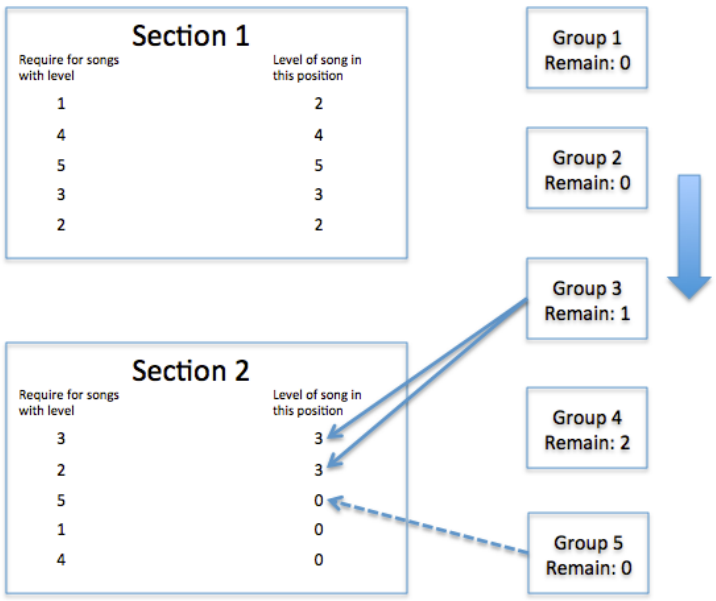


Figure 4.26: Moving a song from higher level group

Referring to Figure 4.26, the third song should be a song with level 5. Group 5 is empty, so the algorithm will find the song in the lower level, which is group 4.

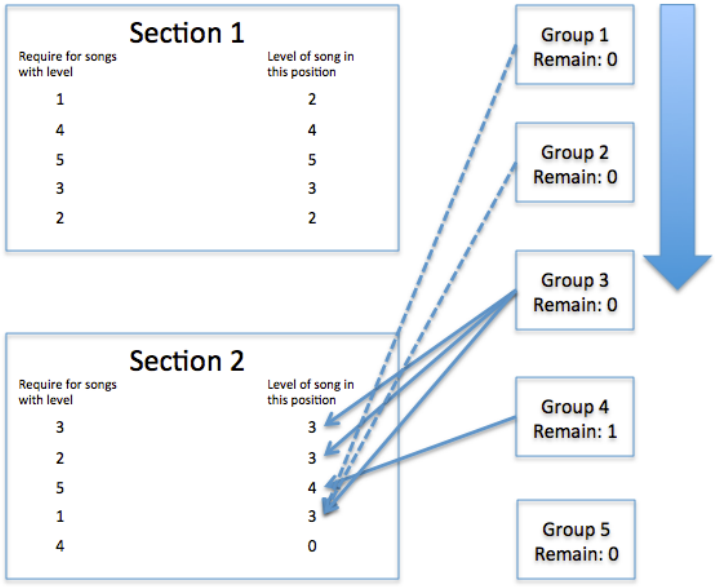


Figure 4.27: The process of finding a song from higher level group

As in Figure 4.27, group 1 is empty when the forth position needs the level 1 song. Because level 1 is the lowest level, the algorithm goes to a greater level to pick the song.

Because group 2 is empty, the algorithm picks a song from group 3 to put in the forth position of the section.

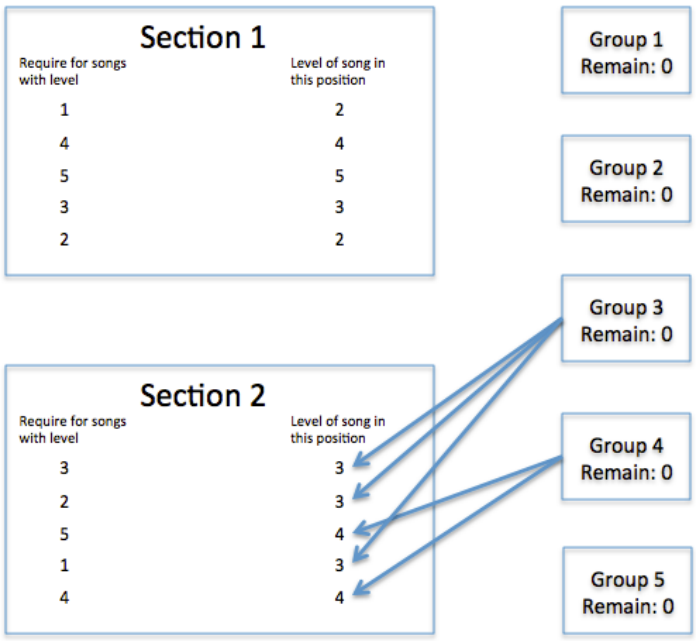


Figure 4.28: Algorithm at termination

As illustrated in Figure 4.28, all songs have been set up. In section 2, the order is 3, 3, 4, 3, and 4. The main code for supervised shuffle can be found in the Appendix 1.

In this shuffle algorithm, we try our best to increase randomness in generating playlist. However, we put the predicted level of the song into the algorithm, and we use that predicted level to supervise the shuffle algorithm. It will make the resulting playlist to be shuffled while also adhering to some rules.

4.5 Simulation Process

We need to test the hybrid engine by using data. The test data are based on specific rules

that were generated based on my own knowledge about the music domain. Although the rules are based on personal perspective, they can create a pattern to test the hybrid engine. The goal was to generate data in which patterns were already known so that the RS/ANN method could be tested to see if the known patterns/rules were indeed discovered. The generation process is as described in Table 4.9

Table 4.9: The simulation process to generating data

Precondition：
Availability of 50 different songs. The songs' genre, album, artist, title will be known to the system and shown to the user.
Pre-operation:
For each song Generate a random integer number between 1 and 5, inclusive. Assign this number as the song's level. Level 5 means the song is the most favored song for this person, and level 1 indicates least favored.
The simulation process:
Step 1: Generate a shuffle song list Step 2: Simulate people listening to the song list based on the song's level For example, if Level of song A is 4, then a number i between 0 and 5 is randomly generated. If i is smaller than Level of song A, we do not skip the song, otherwise, we skip the song. If the i is equal to the song's level, we skip the song. Step 3: System records result of the simulation process into a database. Results include how many times the song has been skipped, and how many times the song has been played and some other information as shown in following tables.

The simulated data represent real users because consider the behavior of one individual listening to music. That user has had many world experiences that contribute to the operations such skip and play he does while listening to music. Take any random

assignment of levels to songs. Convert those levels to operations the user does while listening to music. There is some human being in this world that listens to music in that way. Consider that person listening to 50 different playlists. The person will skip or play according to the level they attach to the songs. The songs he likes are arbitrary. It is random because the experience of real world is random. This is only one potential way to simulate users and other simulation methods may yield different results. However, those methods will simulate one user that exists somewhere in the world. When the simulation process is run again, we just simulate another user that exist somewhere in the world. The only thing we care about is that there is a pattern of one person‘s usage across playlists that contain the same songs in different order.

In Figure 4.29, the song table is used to record the basic information about each song. When we need to analyze by ANN/RS, we mainly use song table’s information. In song table, we record the attributes as describe in Table 4.1 of Chapter 4.

filename	title	artist	genre	album	skiptime	songweight	pcontinue	playtime	scontinue
01 Fever.m4a	01 Fever.m4a	Unknown	Unknown	Unknown	34	2	1	51	0
01 The Best Is...	01 The Best Is...	Unknown	Unknown	Unknown	33	2	4	50	0
03 Georgia On...	03 Georgia On...	Unknown	Unknown	Unknown	8	5	0	53	2
03 Me and Mrs....	03 Me and Mrs....	Unknown	Unknown	Unknown	7	5	4	56	0
Ben E. King - St...	BEN E. KING - S...	Various Artists	Unknown	SENIOR PROM D...	12	5	0	57	1
Beyonce - Halo....	Halo	Beyonce	(14)R&B	WWW.DJCOOL.EU	42	1	0	55	2
Jessie J - Price...	Price Tag (ft. B....	Jessie J	Unknown	rnbmusic.dd.to...	19	3	3	52	0
Jessie J - We Fo...	Jessie J - We Fo...	Unknown	Unknown	Unknown	31	2	0	50	1
Louis Armstron...	What a Wonderf...	Louis Armstrong	Blues	Unknown	6	5	2	51	0
Lucky - Jason...	Lucky (Feat. Col...	Jason Mraz	Pop	We Sing, We Da...	14	5	4	51	0
Maddi Jane - Pr...	Price Tag	Maddi Jane	Pop	Unknown	38	2	0	50	5
Sam Tsui、201...	If I Die Young (...	Sam Tsui	Unknown	Unknown	19	4	2	50	0
The Voice - If I...	If I Ain't Got Yo...	Lily Elise	Pop	Team Christina...	6	5	11	51	0
What are words...	What Are Words	Enn ÊÕ²Ø	Pop	What Are Words...	37	1	1	50	0
世界各地街头艺...	世界各地街头艺...	Unknown	Unknown	Unknown	8	5	8	51	0
周杰伦 - 乌克丽...	周杰伦 - 乌克丽...	Unknown	Unknown	Unknown	39	1	2	50	0
周杰伦 - 傻笑.mp3	傻笑	周杰伦,裘咏琳	Unknown	十二新作	26	3	2	51	0

Figure 4.29: Song Table in database

When the APP is opened, the user picks a song. This is called the user’s first pick song. The table of Figure 4.30 records the user’s first pick song. One of the major operations

users do is to pick the first song. Because we are analyzing the operations users do with the music player, the record in the first pick table is valuable record. In the first pick table, the frequency of appearance of songs is record and is important, because higher frequency indicates the person likes the song. The table of Figure 4.31 records the information about when people skip the song. We record detailed information when the song is skipped. This is also a very important record.

id	filename
1	Beyonce - Halo.mp3
2	Beyond、黄家驹 - 不再犹豫.mp3
3	03 Georgia On My Mind.m4a
4	孙燕姿 - 我不难过.mp3
5	Avril Lavigne - How You Remind Me.mp3
6	梁心颐、周杰伦 - 比较大的大提琴.mp3
7	常石磊 - I Still Believe.mp3
8	陈奕迅、王菲 - 因为爱情.mp3
9	03 Georgia On My Mind.m4a
10	常石磊 - I Still Believe.mp3
11	Sam Tsui、2010 - If I Die Young.mp3
12	玛利亚凯丽 - I Still Believe.mp3
13	Lucky - Jason Mraz&colbie Caillat.mp3
14	03 Georgia On My Mind.m4a
15	梁心颐、周杰伦 - 比较大的大提琴.mp3
16	陈奕迅 - 一丝不挂.mp3
17	4 In Love - 命中注定.mp3

Figure 4.30: Table Firstpick in database

id	filename	position	amount	pcontinue	scontinue
1,638	Avril Lavigne -...	46	50	25	0
2,261	Avril Lavigne -...	38	50	23	0
1,404	周杰伦 - 爱你没...	41	50	20	0
2,347	What are words...	10	50	18	0
1,244	周杰伦 - 乌克丽...	43	50	17	0
742	Avril Lavigne -...	36	50	15	0
775	周杰伦 - 乌克丽...	42	50	14	0
353	周杰伦 - 乌克丽...	8	50	13	0
1,005	周杰伦 - 红尘客...	27	50	13	0
2,267	Beyond、黄家驹...	0	50	13	0
606	01 Fever.m4a	32	50	12	0
1,018	Beyond、黄家驹...	47	50	12	0
1,354	陈奕迅 - 淘汰.mp3	1	50	12	0
1,752	周杰伦 - 爱你没...	17	50	12	0
1,787	周杰伦 - 乌克丽...	33	50	12	0
2,068	01 Fever.m4a	25	50	12	0
2,452	Jessie J - Price...	9	50	12	0

Figure 4.31: Table Skipinfor in database

The descriptions of each column in the tables of Figure 4.29 and 4.30 were given in

Tables 4.2 and 4.3 in the previous chapter.

The main point of the simulation process is to generate data that reflect how people listen to music. In this process, we assume that people will skip songs based on how much they like them. The level of the song measures how much the user likes the song. The way the level is computed has been previously described in Table 4.9. If the song is high level for a person, the chance to skip this song is lower than if the person does not favor the song. This simulation data cannot only be used as training set for ANN, but also it can be used as testing data.

There is a simulation function in the system as illustrated in Table 4.9. In the input field, we can input how many playlists we want to simulate. Referring to Figure 4.32, we input 50 into the input field to indicate that we wish to simulate 50 playlists. Clicking the simulate button gets us our simulation data. In the simulated data, each song has its own level initially generated randomly. Moreover, all the songs are first put into a shuffle playlist, and then the listening process is simulated. The shuffle playlist is the song list that is shuffled by the Modern Shuffle algorithm as we described in Subsection 2.1.1. This step is repeated 50 times according to Figure 4.32. But repeating the step 100 times was done, and, in fact, the simulation process can be run to generate as many playlist as the desired.

The 50 times simulation process can also be treated as a person listening to songs in 50

different playlists. In one playlist, each song is only played 1 time. For each of these songs, we record the number of times it is played, the number of times it is skipped, the continually play times and continually skip times in simulation process. As we mentioned in database design, the record of continually play times can reveal whether this song is favorite for this user, and the record of continually skip times can reveal whether this song has started to become boring for this user. If the song in 50 different playlists was played continually 30 times, it is enough evidence to assume this song is the user's favorite song. We also record the skip times and play times. The ratio between play times and skip times can reveal the song's level by looking on the big picture instead of details. So, the whole simulation process can be interpreted as a human who is listening to music and he will choose whether to skip or play a song based on the preference of the song (level). The system will record every skip and play operation of this virtual human.

Figure 4.32: UI for developer

If we click the simulation again, all the songs' levels will be randomly assigned to new levels randomly, and the system will do the simulation process again starting from the beginning. We will get a totally different data set by clicking the simulate button. By this means, we get data to train the ANN, and by the same method we get data to test the ANN.

There are limitations of our simulation process. We simulate the process of people listening to music by assuming every song has its own level in users' minds. Then, users decide whether they will skip it based on the song's level. But, in the real world, users sometimes want to listen to a song when they are in a special mood even if this song is not at a high level in their mind. For example, a user may pick a song he is not heard in a long time even though it is not his favorite. There are some other unpredictable movements of the users such as hitting the wrong button or getting interrupted that are not captured by our simulation process. So, the simulation data cannot fully reveal the situation of real people listening to music. The pattern in real data may be more complicated, and the hybrid engine may not be able to learn the pattern. However, there are patterns in the simulation data that represent some features of the listening process. Once the hybrid engine can process the simulation data and be able to learn the predetermined pattern, it is likely that the hybrid engine is able to learn the patterns in other data including the real human preference data.

4.6 The future work

After the APP is published in the APP store, we need to get feedback from users for the future work. There is a function in APP that allows users to send an email that contains their own database and comments to our email account. The method of acquiring data from users is illustrated as Figure 4.29.

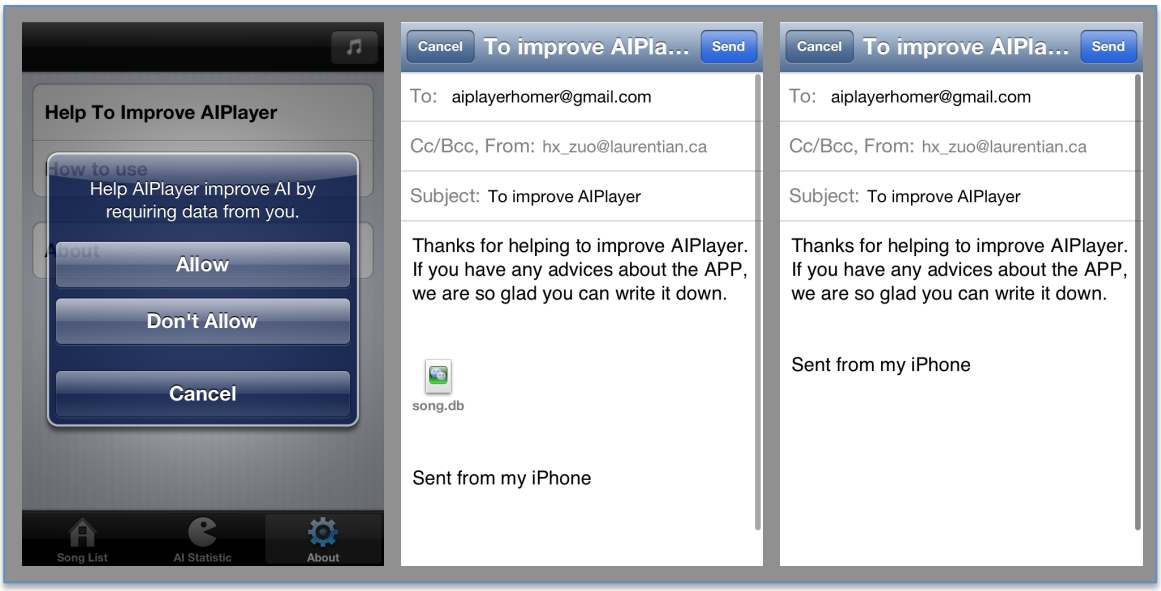


Figure 4.33: UI of User Feed Back

In Figure 4.33, when the user clicks the option Help To Improve AIPlayer, the system will pop up an alert window that acquires the approval from the user to send the database for feed back. If user clicks the Allow button, the content of email will include the database, otherwise it does not. As in Figure 4.33, the middle picture is the email that includes the database; the right picture is an email that only has words.

By collecting user's feed back, we can get a lot of other people's idea about the

self-learning function. In the future work, if users can send their database to us, the data in those databases can show how well the hybrid engine is working in user's iPhone. In this way, they can help to improve the system.

In this chapter, we have explained the purpose of major design decisions in the project and outlined the solution of the problem as was presented in a former chapter. In particular, we showed how we built the hybrid engine, and made it portable. We showed how we created a friendly UI for the user. The evaluations of the implementation especially regarding hybridization are shown in the next chapter.

Chapter 5

5 Evaluation

5.1 Tool for Evaluation

In order to evaluate the system, we construct a developer UI for developer to test the system and evaluate the results. Every button has its own function, and the result will be displayed in the console window of XCode, which is Integrated Development Environment (IDE) of objective C programming language. The developer UI is as illustrated in Figure 5.1.

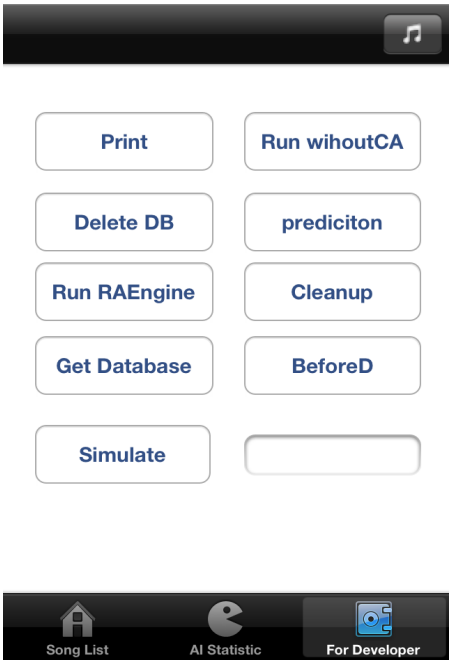


Figure 5.1: Developer UI

As illustrated in Figure 5.1, there are 9 buttons and an input field.

1. Button Print is used to print the data in the Console window of XCode.

2. Button Delete DB is used to delete the database file.

3. Button Run RAEngine is used to run the hybrid engine. When the developer clicks this button, the hybrid engine will be activated and the results will show up in the console window such as the result of rough set by itself, the information about the ANN before, during and after training.

4. Button Get Database provides a function to withdraw the database file from the APP.

5. Button Run withoutCA is used to run the ANN. By clicking this button, the system only actives the ANN engine. After running, the console window will display the information about ANN before training, the iterations of the training process and the information about ANN after training.

6. Button Prediction provides evaluation of the ability of ANN's prediction. When developer clicks this button, the prediction result, the desire result and the summary about the error between prediction and desire result will be displayed in the console window.

7. Button Clean Up provides the developer with a method to delete all the data other than the base song information in the song table.

8. Button BeforeD will generate dsong table without doing discretization method.

9. Button simulation can be used to generate simulation data according to a number the developer input in the input field. The detail of simulation process will be presented in the next sections.

Some of the testing result is displayed in Appendix 3.

5.2 Assessment of Artificial Neural Network

By correctness of NN, we mean the NN will make a good prediction after training. A good prediction is defined as small error between desired prediction and actual prediction. Two methods will be described for assessing quality of the NN hybrid Engine. The first method was intended to show the precision of prediction by using trained NN with the help of RST. In fact, the precision of prediction did not improve with addition with RST. The second method was to show that the training process is more stable when using the hybrid engine than when using only the NN engine. Stability means that the number of iterations used to train is about the same for every training run.

Method 1: Check the accuracy of ANN's prediction by using different numbers of shuffle playlists used in the simulation

Step 1: The system generates data from 100 playlists by simulation.

Step 2: By using the simulated data, we train the ANN and record after training, the weights and threshold value into the database.

Step 3: The system cleans up the simulated data, and repeats the simulation process with difference number of iteration in the simulation process.

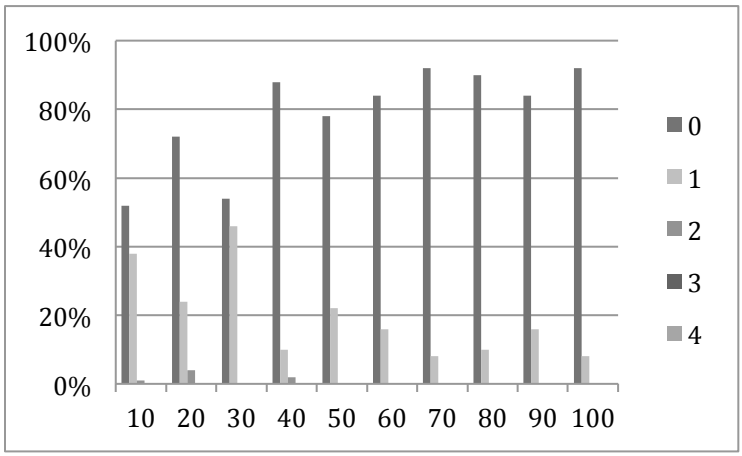
Step 4: Use trained ANN to predict the levels of a song by using the new data from step 3.

The absolute error between predicted value and desired value are analyzed in Table 5.1. This table is based on 50 songs. The numbers shown in third to seventh columns is the number of songs. For example, in first row and third column, the number is 26, so that it is 52% in 50 songs.

The absolute error value of prediction =| Prediction value – Desired value|.

Table 5.1: Evaluation of ANN for different simulation sizes

Serial number	Times of Simulation Process	The absolute error value of prediction				
		No error	1	2	3	4
1	10	26(52%)	19(38%)	5(10%)	0	0
2	20	36(72%)	12(24%)	2(4%)	0	0
3	30	27(54%)	23(46%)	0	0	0
4	40	44(88%)	5(10%)	1(2%)	0	0
5	50	39(78%)	11(22%)	0	0	0
6	60	42(84%)	8(16%)	0	0	0
7	70	46(92%)	4(8%)	0	0	0
8	80	45(90%)	5(10%)	0	0	0
9	90	42(84%)	8(16%)	0	0	0
10	100	46(92%)	4(8%)	0	0	0



From Table 5.1, we can see that the ANN engine predicts with fewer errors, with increasing number of iterations of the simulation process. For example, in row 7, there are 46 songs out of 50 with no error when we have simulation times of 70. When the sample data are small, the accuracy of ANN's prediction is unstable as shown in rows 1, 2 and 3 of Table 5.1.

Recall that song level is a measure of how much the user likes this song. For example, if the levels of a song are 5, but the prediction is 1, then the error will be 4. Errors of magnitude 4 are shown in column 7. In Table 5.1, we can see that no errors of such magnitude occurred.

Table 5.2 shows the result of random prediction. By random prediction, we mean that we randomly generate a number from 1 to 5 for each song's prediction level, and then we compare it with the desired levels of a song. We can compare the data between Table 5.1 and Table 5.2. By observing the data on Table 5.2, the precision of random prediction is low, because there are only 20% of the predictions with no error. Further more, the error

is distributed in the high error rate column.

Table 5.2: Result of random prediction

Serial number	Times of Simulation Process	The absolute error value of prediction				
		No error	1	2	3	4
1	10	12(18%)	13(28%)	12(28%)	9(18%)	4(8%)
2	20	8 (16%)	21(42%)	8(16%)	11(22%)	2(4%)
3	30	15(30%)	15(30%)	8(16%)	7(14%)	5(10%)
4	40	9(18%)	18(36%)	7(14%)	12(24%)	4(8%)
5	50	10(20%)	14(28%)	14(28%)	7(14%)	5(10%)
6	60	11(22%)	16(32%)	12(24%)	5(10%)	6(12%)
7	70	13(26%)	14(28%)	14(28%)	6(12%)	3(6%)
8	80	8(18%)	14(28%)	16(32%)	9(18%)	3(6%)
9	90	6(12%)	18(36%)	12(24%)	7(14%)	7(14%)
10	100	16(32%)	13(26%)	12(24%)	4(8%)	5(10%)

From Table 5.1, we can see that the entries are mostly placed on column 3 and column 4, and that there are no elements on columns 6 and 7. In contrast, in Table 5.2, we can see that the entries are mainly in columns 3 and 4. From the fact that the numbers are in different columns, we can conclude the error in Table 5.1 is less, because the absolute value of error between predicted and desired is less. Observe also that the entries in Table 5.2 columns 3 and 4 are always more than 50%. This means that at least there are 50% of random predictions have error. Moreover, around 25% of entries are on columns 6 and 7. There are 25% of random predictions have big error. Compare with random prediction, we can see that the prediction by using hybrid engine is more precise.

Method 2: Check the accuracy by using the same simulation process but with simulation times constant at 70.

Step 1: Use the same ANN generated by Method 1's step 2

Step 2: The system deletes all the simulated data, and redoes the simulation process, setting the number of times to 70. The reason to choose 70 is that the accuracy is highest in Table 5.1 at a simulation time of 70 as observed in row 7.

Step 3: Use ANN to predict the levels of songs and observe the error.

Step 4: Redo Steps 2 and 3 for 10 times to keep the table size small so that it can be presented on a page. Observe the result.

Table 5.3: Evaluation of ANN with same simulation

Serial number	Times of Simulation Process	The absolute error value of prediction				
		No error	1	2	3	4
1	70	42(84%)	8(16%)	0	0	0
2	70	44(88%)	6(12%)	0	0	0
3	70	41(82%)	9(18%)	0	0	0
4	70	45(88%)	5(10%)	0	0	0
5	70	43(86%)	6(12%)	1(2%)	0	0
6	70	45(90%)	5(10%)	0	0	0
7	70	43(86%)	7(14%)	0	0	0
8	70	44(88%)	5(10%)	1(2%)	0	0
9	70	41(82%)	8(16%)	1(2%)	0	0
10	70	36(72%)	14(28%)	0	0	0

According to method 2, we picked 70 as the number of times to execute in the simulation process to generate the data as observed in Table 5.3. Each row of the table describes a completely different simulation. The prediction of ANN is working well. There are always greater than 72% of predictions that have no error. For the others that have error,

the absolute error stays mostly at a magnitude of 1. Moreover the prediction is stable, because we can see that there are no absolute errors of prediction of magnitude 3 or 4. Because the level of the song is randomly generated in simulation data, the absolute error value has chances that it can never be able to reach 3 or 4. For example, in order to make the absolute error value to 4, the desired value has to be 5 at the same time the predicted value is 1. What if the level of all songs in the playlist is randomly generated to be 3? The largest absolute error value can only be 2. However, we can observe that the value of the fifth column is only around 2%. Even if in the fourth column, the value is way less than that of the third column. We can conclude that the prediction function of NN engine is working well.

In these 2 methods, the value of weights in each neuron of NN is illustrated in Figure 5.2. Training was accomplished using method 1. The system only trained ANN one time by using the sample data, but the accuracy of NN prediction is really good. In Table 5.1 and Table 5.3, they all show that the accuracy of prediction is always greater then 50%. The others that have error are all only one absolute error. It means that the NN prediction is working in different situations.

Figure 5.2 is the outcome of NN displaying weights and threshold values for all the neurons in neural network. Each neuron is an information-processing unit. Multiple neurons in the hidden layer as well as the output layer are collaborating with each other to discover patterns in data.

Hidden layer:

Neuron 1:

The weights is:

W0: -0.718099....W1: 0.187835....W2: 0.371021....W3: -0.806052....W4: 0.787110....W5: -0.075547....W6: 0.411714....W7: 0.492584....W8: -0.006649....W9: 0.246519....W10: -0.281735.... Threshold: -0.000694

Neuron 2:

The weights is:

W0: 0.001489....W1: -0.170555....W2: 0.205392....W3: -0.026753....W4: 0.124791....W5: -0.096923....W6: 0.158530....W7: 0.273285....W8: -0.169372....W9: 0.245434....W10: 0.437853.... Threshold: 0.065311

Neuron 3:

The weights is:

W0: -0.007148....W1: -0.163889....W2: 0.215458....W3: -0.031910....W4: 0.135221....W5: -0.099302....W6: 0.169442....W7: 0.278275....W8: -0.166367....W9: 0.251020....W10: 0.429216.... Threshold: 0.154087

Neuron 4:

The weights is:

W0: -0.626821....W1: 0.055998....W2: 0.112527....W3: -0.438380....W4: 0.273440....W5: 0.044715....W6: 0.083401....W7: 0.398355....W8: -0.099332....W9: 0.337179....W10: -0.190457.... Threshold: 0.139346

Neuron 5:

The weights is:

W0: -0.023707....W1: -0.117903....W2: 0.264907....W3: -0.034671....W4: 0.178118....W5: -0.081758....W6: 0.209264....W7: 0.313921....W8: -0.141448....W9: 0.281939....W10: 0.412657.... Threshold: -0.229510

Neuron 6:

The weights is:

W0: -0.904835....W1: -0.039327....W2: 0.099243....W3: -0.629220....W4: 0.417192....W5: -0.182820....W6: 0.166633....W7: 0.336826....W8: -0.266705....W9: 0.024657....W10: -0.468472.... Threshold: -0.136353

Output Layer:

Neuron 1:

The weights is:

W0: -2.146710....W1: 0.964392....W2: 0.769011....W3: -1.307864....W4: 0.899228....W5: -1.575448.... Threshold: -0.762298

Figure 5.2: The outcome of ANN

5.3 Assessment of Hybrid System with Rough Set and Artificial Neural Network

By using Rough Sets we core the data, which means we come to know which condition attributes are core attributes. None of the core attributes are redundant meaning that any additional attribute will provide no additional classification power. The core data are applied to ANN to make it converge more quickly. Two methods of evaluating the hybridization are discussed next.

Method 1: Comparison of ANN hybrid RS engine and ANN engine by itself with fixed size simulation data.

Step 1: The system does the simulate process with fixed size 100 times playlists.

Step 2: By using this simulation data, we train the two engines and see how many iterations each of them needs.

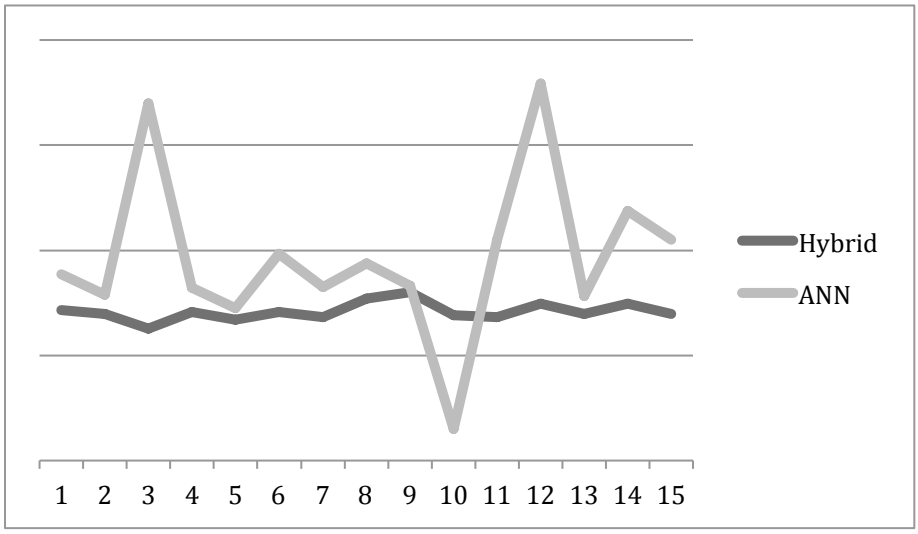
Step 3: go back to step 2 and do that again. Check the stability of the engines.

Table 5.4 indicates that ANN hybrid RS stabilizes with less iterations than ANN by itself and produces a lower error rate. This can be explained as follows: In ANN engine, weights are initialized randomly. That is the reason why ANN engine is unstable. When the randomly initialized weights are so far from what we really need, ANN engine will take a long time to converge. However, with ANN hybrid RS engine, the RS engine will

control the initialization and weight training process of ANN by using core characteristics.

Table 5.4: Comparison of ANN hybrid RS and ANN with fixed size simulation data

Serial number	Times of Simulation Process	ANN hybrid RS		ANN	
		Number of Iteration	Errors	Number of Iteration	Errors
1	100	27	0.000003	59	0.000009
2	100	25	0.000001	38	0.000010
3	100	18	0.000000	2486	0.000010
4	100	26	0.000001	44	0.000009
5	100	22	0.000005	28	0.000008
6	100	26	0.000001	93	0.000010
7	100	23	0.000000	45	0.000010
8	100	35	0.000008	75	0.000008
9	100	40	0.000002	46	0.000006
10	100	24	0.000007	2	0.000005
11	100	23	0.000009	126	0.000010
12	100	31	0.000006	3852	0.000010
13	100	25	0.000001	37	0.000003
14	100	31	0.000000	236	0.000010
15	100	25	0.000002	1275	0.000010



In ANN hybrid RS engine, the weight of any unnecessary condition attributes will be initialized to 0, if it is the first time to run the engine. And the learning rates of the weight

of these condition attributes are also set to 0. It means that no changes to such weights are made when the engine does weight training. This means that unnecessary condition attributes will have no impact on the weight training process. The chance that the system needs to converge with worse initial weights is reduced with fewer attributes. It will improve the stability of the system. The stability was defined at Subsection 4.2.3. The stability can influent the performance of the APP. The hybrid engine is going to be active when the APP is working. The stability of hybrid engine will ensure the stability of the APP.

In method 2, we generate different sizes of simulation data, and use ANN hybrid RS and ANN by itself to do sample data training. We compare the number of iterations and errors after convergence between the two engines. But we set the maximum iteration to 10000 after which the engine will stop attempting to reach convergence. In such case, we can nevertheless compare the errors to see which engine is converging more quickly.

Method 2: Compare performance between ANN hybrid RS engine and ANN engine with different size simulation data

Step 1: The system does different times simulation process.

Step 2: By using this simulation data, we train the two engines and see how much iteration do they each need.

Step 3: Go back to step 1 and do that again. Check the number of iteration and the

convergence error.

In the Table 5.5, number of iterations means the how many times the ANN has to go though the test data. The details of that can be checked in multiple layer neural networks presented in chapter 2.

Table 5.5: Comparison of ANN hybrid RS and ANN with different size simulation data

Serial number	Times of Simulation Process	ANN hybrid RS		ANN	
		Number of Iteration	Errors	Number of Iteration	Errors
1	10	10001	0.000616	2700	0.000010
2	20	10001	0.004127	10001	0.003309
3	30	11	0.000000	53	0.000004
4	40	64	0.000009	19	0.000001
5	50	1	0.000000	230	0.000010
6	60	10001	0.000118	10001	0.000029
7	70	332	0.000010	434	0.000010
8	80	2073	0.000010	19	0.000002
9	90	7379	0.000010	1638	0.000010
10	100	10001	0.001245	10001	0.000431

According to Table 5.5, it is hard to tell the advantage of using ANN with RS engine. By different simulation data, the performance of ANN hybrid RS engine is unstable, or we can say the stability depends on the data. Sometimes ANN hybrid RS engine works way better than ANN engine, and sometimes not. The reason is that the randomly initialized weights play a very important part in determining how long the learning process will take. That is why sometimes using randomly initialized weight method is better than supervised randomly initialized process.

In the cellphone software, we cannot have the engine running for too many iterations. So,

we set a boundary on the number of iterations. When the number of iterations hits the boundary, the engine will stop weight training and record the ANN. Referring to following Table 5.6, the accuracy of prediction is good. Observe from the table that some of the ANN does not fully converge.

Method 3: To see whether lack of full convergence will affect the prediction, we use 5 steps:

Step 1: The system does different times simulation process.

Step 2: By using this simulation data, we train ANN hybrid RS engine.

Step 3: When the number of iterations hits 500, the engine stops weight training.

Step 4: Record the ANN into the database. Use it to make a prediction, and check the accuracy.

Step 5: Go back to step 1 and do this method again, and observe the absolute error value of prediction.

According to Table 5.6, we can see that, lack of ANN convergence does not affect the accuracy of predictions very much. In the table, the 1, 4, 6, 10 rows show this conclusion. When the ANN fully converges, the prediction is precise. However, even when the ANN does not fully converge, prediction is not entirely wrong.

Table 5.6: Accuracy of prediction when ANN is not fully convergent

	Times of Simulation Process	ANN hybrid RS		The absolute error value of prediction				
		Number of Iteration	Errors	Zero	1	2	3	4
1	10	500	0.000035	32(64%)	18(36%)	0	0	0
2	20	19	0.000000	32(64%)	17(34%)	1(2%)	0	0
3	30	26	0.000008	31(62%)	19(38%)	0	0	0
4	40	500	0.014690	44(88%)	6(12%)	0	0	0
5	50	13	0.000000	28(56%)	22(44%)	0	0	0
6	60	500	0.000179	37(74%)	13(26%)	0	0	0
7	70	55	0.000008	33(66%)	17(34%)	0	0	0
8	80	78	0.000010	41(82%)	9(18%)	0	0	0
9	90	31	0.000007	32(64%)	18(36%)	0	0	0
10	100	500	0.001398	47(94%)	3(6%)	0	0	0

Thus making a boundary for iteration due to practical matters appears to be reasonable. In future work, we will do the ANN self-learning again based on a timer. So, the ANN will perfect itself again and again. With more research, it is expected that the ANN will incrementally move toward better simulation of peoples' thinking in this area, and ultimately make the same decisions as people.

Chapter 6

6 Conclusion

In this project, we built a hybrid engine to learn user preferences for music. The engine is based on a multilayer neural net to make the learning process more effective. Rough Set (RS) is a method to analysis data are characterized by imprecision, inconsistency and incompleteness. RS was used during the learning process to help train the weight of the neural net. We evaluated the engine by using simulation data. The simulation data are generated based on specific rules, and the hybrid engine was able to learn the patterns input to data by those rules. The hybrid engine was designed to be portable, so that it can be used not only in this project, but also to solve other real life problems.

We are still perfecting the user interface of the APP. We have applied to the APP store for publication of the APP that has been accepted, and people are currently using it. The future work is to let users decide whether this engine is good based on their experience with the APP, and we can improve the APP by consider the feedback.

The hybrid engine is portable. It has been designed so that it can be used on different data sets. The engine provides an effective way to run artificial neural network and rough set engine separately or in combination. When other researchers and other projects need a software library to solve some similar problem, our engine will be very useful and also easy to modify. We have post the class library and an instruction of the library on the web

for public use.

This project reveals a new way for predicting users' favored playlists. It is not like the iTunes of Apple Company. We do not have a big database and big server to analyze all customers' data and draw the conclusions. In contrast, we focus on analysis of each specific user. Each APP has its own database. The APP needs to learn the user habits based only on this particular user's operations. Consequently, every user has his own specific ANN for predicting their favorite songs. Each hybrid engine will be custom tailored for each user. The more users use the APP, the more the APP will fit them.

In artificial neural network hybrid rough set domain, we have described implementation details of the hybridization in detail. In a lot of papers [4][3], the authors are using some third-party software to combine RS and ANN. However, in this thesis, we implement the RS and ANN engine in its entirety in objective C, so we can do the hybridization at the coding level. We use rough set's core characteristics to guide ANN's learning process in the code. The result shows that this hybrid approach improves the stability of training the ANN.

To summarize the findings, the hybrid engine makes more precise prediction than randomly predicting the levels of a song. There are some limitations because we are using simulation data to test the hybrid engine. However, it still shows that the hybrid engine is capable to deal with certain patterns that were previously put in the data. Generally, the

hybrid engine in which insignificant attributes are not considered stabilizes the training process more than NN by itself in which weights of all attributes are initialized randomly.

References

[1] L. Barrington, R. Oda, and G. Lanckriet, "Smarter than genius? human evaluation of music recommender systems," ... *Symposium on Music ...*, 2009.

[2] A. Watson, "The world according to iTunes: mapping urban networks of music production," *Global Networks*, vol. 12, no. 4, pp. 446–466, Oct. 2012.

[3] C.-L. Chuang and S.-T. Huang, "A hybrid neural network approach for credit scoring," *Expert Systems*, vol. 28, no. 2, pp. 185–196, May 2011.

[4] Y. Shen, T. Li, E. Hermans, D. Ruan, G. Wets, K. Vanhoof, and T. Brijs, "A hybrid system of neural networks and rough sets for road safety performance indicators," *Soft Computing*, vol. 14, no. 12, pp. 1255–1263, Sep. 2009.

[5] P. Zdzislaw, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[6] H. S. Own and A. Abraham, "A new weighted rough set framework based classification for Egyptian NeoNatal Jaundice," *Applied Soft Computing*, vol. 12, no. 3, pp. 999–1005, Mar. 2012.

[7] R. Bello and J. L. Verdegay, "Rough sets in the Soft Computing environment," *Information Sciences*, vol. 212, pp. 1–14, Dec. 2012.

[8] S. K. M. WONG and ZIARKO W., "Algorithm for inductive learning," *Bulletin of Polish Academy of Sciences*, vol. 34, pp. 271–276, 1986.

[9] Y.-S. Chen and C.-H. Cheng, "Hybrid models based on rough set classifiers for setting credit rating decision rules in the global banking industry," *Knowledge-Based Systems*, vol. 39, pp. 224–239, Feb. 2013.

[10] C.-C. Yeh, F. Lin, and C.-Y. Hsu, "A hybrid KMV model, random forests and rough set theory approach for credit rating," *Knowledge-Based Systems*, vol. 33, pp. 166–172, Sep. 2012.

[11] Y. F. Hassan, "Rough sets for adapting wavelet neural networks as a new classifier system," *Applied Intelligence*, vol. 35, no. 2, pp. 260–268, Sep. 2010.

[12] R. B. Bhatt and M. Gopal, “On fuzzy-rough sets approach to feature selection,” *Pattern Recognition Letters*, vol. 26, no. 7, pp. 965–975, May 2005.

[13] S. K. M. Wong and W. Ziarko, “On learning and evaluation of decision rules in the context of rough sets,” in *Proceedings of the ACM SIGART international symposium on Methodologies for intelligent systems* -, 1986, pp. 308–324.

[14] S. K. M. Wong, W. Ziarko, and R. L. Ye, “Comparison of rough-set and statistical methods in inductive learning,” *International Journal of Man-Machine Studies*, vol. 25, no. 1, pp. 53–72, Jul. 1986.

[15] G. Zhang, B. Eddy Patuwo, and M. Y. Hu, “Forecasting with artificial neural networks:,” *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, Mar. 1998.

[16] R. Andreani, E. H. Fukuda, and P. J. S. Silva, “A Gauss–Newton Approach for Solving Constrained Optimization Problems Using Differentiable Exact Penalties,” *Journal of Optimization Theory and Applications*, vol. 156, no. 2, pp. 417–449, Jun. 2012.

[17] Y. Zhang, D. Li, X. Fu, and W. Bi, “An improved Levenberg–Marquardt algorithm for extracting the features of Brillouin scattering spectrum,” *Measurement Science and Technology*, vol. 24, no. 1, p. 015204, Jan. 2013.

[18] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the Marquardt algorithm.,” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 5, no. 6, pp. 989–93, Jan. 1994.

[19] M. Negnevitsky, “Artificial neural networks,” in in *Artificial Intelligence: A Guide To Intelligent Systems*, Second., Addison-Wesley, 2005, pp. 165–217.

[20] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.,” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[21] L.-M. Fu, *Neural Networks In Computer Intelligence*. Singapore: McGraw-Hill Education, 1994.

[22] S. Haykin, *Neural networks : a comprehensive foundation*, 2nd ed. Prentice Hall, 1999.

[23] A. Øhrn, J. Komorowski, A. Skowron, and P. Synak, “The design and implementation of a knowledge discovery toolkit based on rough sets-The ROSETTA system,” 1998.

[24] S. Chandana and R. V. Mayorga, “Rough Neuron based on Pattern Space Partitioning,” *Neurocomputing*, vol. 74, no. 12–13, pp. 2052–2061, Jun. 2011.

[25] A. Ganivada, S. Dutta, and S. K. Pal, “Fuzzy rough granular neural networks, fuzzy granules, and classification,” *Theoretical Computer Science*, vol. 412, no. 42, pp. 5834–5853, Sep. 2011.

[26] S. Kotsiantis and D. Kanellopoulos, “Discretization techniques: A recent survey,” *GESTS International Transactions …*, vol. 32, no. 1, pp. 47–58, 2006.

[27] S. Gottlieb, C.-W. Shu, and E. Tadmor, “Strong Stability-Preserving High-Order Time Discretization Methods,” *SIAM Review*, vol. 43, no. 1, pp. 89–112, Jan. 2001.

[28] Y. Yang and G. Webb, “A comparative study of discretization methods for naive-bayes classifiers,” *Proceedings of PKAW*, pp. 159–173, 2002.

[29] E. Frank and I. Witten, *Data mining: practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[30] D. Andrés and L. Pérez, “Efficient Parallel Random Rearrange,” *… Symposium on Distributed Computing and Artificial …*, pp. 183–190, 2011.

[31] H. Mads, “Random.org.” [Online]. Available: http://www.random.org.

[32] R. . Fisher and F. Yates, *Statistical tables for biological, agricultural and medical research*, 3rd ed. London: Oliver & Boyd, 1948, pp. 26–27.

[33] R. Durstenfeld, “Algorithm 235: random permutation,” *Communications of the ACM*, vol. 6, pp. 1963–1964, 1964.

[34] A. Atiya and C. Ji, “How initial conditions affect generalization performance in large networks.,” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 8, no. 2, pp. 448–51, Jan. 1997.

[35] D. R. Hipp, “SQLite.” [Online]. Available: http://sqlite.org.

Appendix

A1. The Important code

The code of Supervise Shuffle algorithm

```
+(void)DJShuffle:(NSMutableArray *)songList :(Song *)song
{

    int section = 5;//Define how many songs in one section
    int count = [songList count];
    int sign = 0;
    BOOL result = NO;
    int * indexes = (int *) malloc(5 * sizeof(int));
    NSLog(@"count: %d",count);
    NSMutableArray * group = [[NSMutableArray alloc]init];
    //split songs into different group based on their level
    for (int i = 0; i < section; i ++) {
        [group addObject:[self groupTheList:songList :i+1]];
        NSLog(@"i : %d, count : %d",i , [[group objectAtIndex:i ] count]);
    }
    int debugC = 1;
    while (YES) {
        NSLog(@"\necho:%d",debugC);
        sign = 0;
        //generate the order of level in one section
        [self generateRandomIndex:indexes :section];
        for (int i = section-1; i >= 0; i --) {
            int n = indexes[i];

            NSLog(@"n : %d",n);
            //randomly pick a song from specific group
            if (![self randomPickFromGroup:songList :[group objectAtIndex:n]]) {
                //pick a song from the other group
                result = [self pickFromTheOtherGroup:section :n :group :songList];
```

```
                }else
                {
                    NSLog(@"get it in: %d",n);
                }
            }
            if(result){
                NSLog(@"it run out!!!!!!!!!!!");
                break;
            }
            debugC++;
        }
        count = [songList count];
        NSLog(@"count: %d",count);

        [group release];
        free(indexes);
}
```

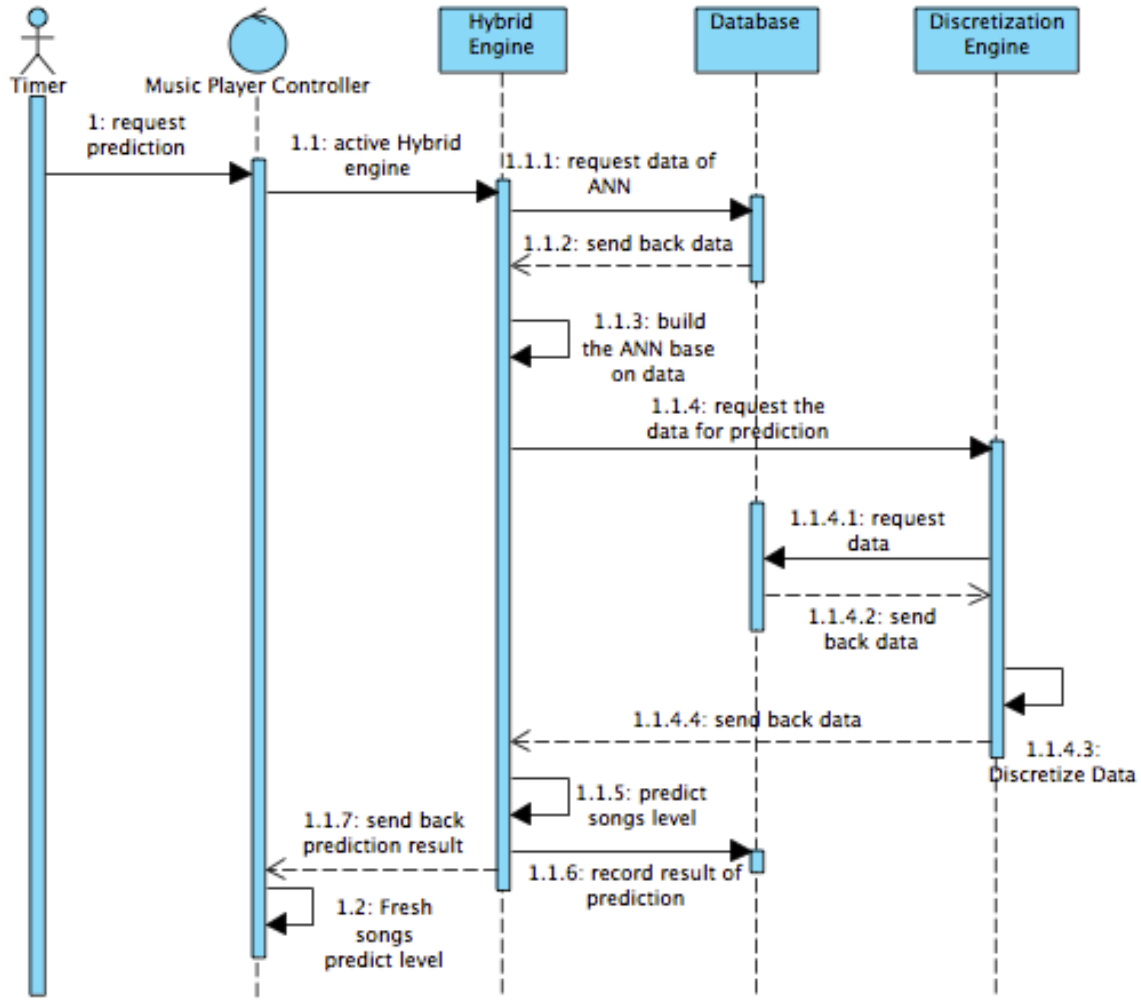
The main code of ANN engine

```
-(void)doTraining
{

    int iteration = 0;
    double errorsign = 0;
    NSMutableArray * hiddenLayer = [ann getHiddenLayer];
    NSMutableArray * outputLayer = [ann getOutputLayer];
    NSMutableArray * inputdataArray = [inputArray getDataArray];
    int dataCount = [inputdataArray count];

    for (; ; iteration++) {

        for (int i = 0; i < dataCount; i++) {
            NSObject<NNInputObject> * tempdata = [inputdataArray objectAtIndex:i];
            for (int j = 0; j < numHidden; j++) {
                ANeuron * tempNeuron = [hiddenLayer objectAtIndex:j];
                [tempNeuron caculateOutput2:tempdata];
            }
            for (int k = 0; k < numOutput; k++) {
```

```
                    ANeuron * tempNeuron =    [outputLayer objectAtIndex:k];
                    [tempNeuron caculateOutput1:hiddenLayer];
                }
                errorsign = [self weightTraining:tempdata :hiddenLayer :outputLayer];
            }
            if (errorsign <= error)
                break;
            else if (iteration > 500)
                break;
            else
            {
                errorsign = 0;
                continue;
            }
        }
        NSLog(@"iteration: %d", iteration);
        NSLog(@"error: %f",errorsign);
        [ann recordANN];

}

-(void)predict
{
        //////////////////////
        int zero = 0;
        int one = 0;
        int two = 0;
        int three = 0;
        int four = 0;
        int error = 0;
        //////////////////////
        NSMutableArray * hiddenLayer = [ann getHiddenLayer];
        NSMutableArray * outputLayer = [ann getOutputLayer];
        NSMutableArray * inputdataArray = [inputArray getDataArray];
        int dataCount = [inputdataArray count];
        for (int i = 0; i < dataCount; i++) {
            NSObject<NNInputObject> * tempdata = [inputdataArray objectAtIndex:i];
            [tempdata printself];
            for (int j = 0; j < numHidden; j++) {
                ANeuron * tempNeuron = [hiddenLayer objectAtIndex:j];
                [tempNeuron caculateOutput2:tempdata];
```

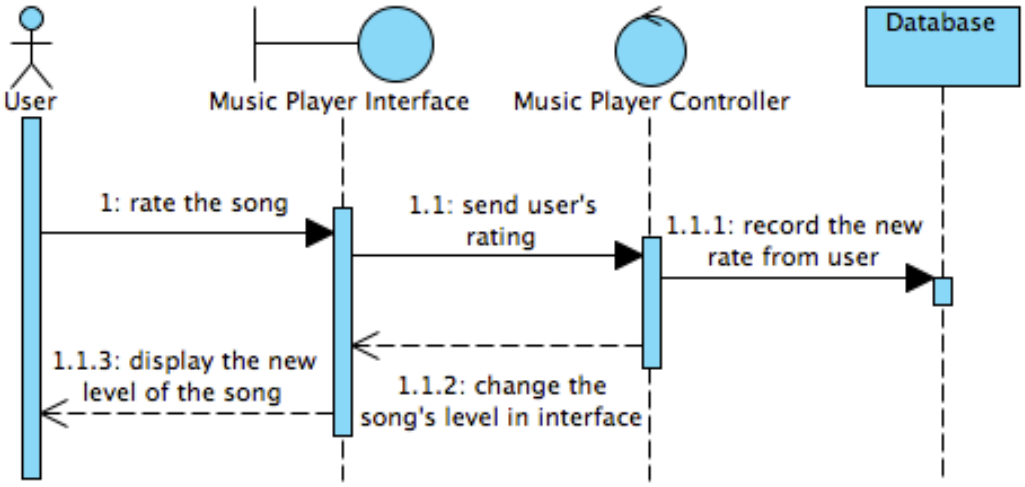
```
            }
            for (int k = 0; k < numOutput; k++) {
                ANeuron * tempNeuron =    [outputLayer objectAtIndex:k];
                [tempNeuron caculateOutput1:hiddenLayer];
                double result = (int)([tempNeuron getOutput]*5+0.5);
                [ANNUtility updatePredictRate:tempdata :result];
                NSLog(@"desire: %d",(int)([tempdata getdesire:k]*5));
                NSLog(@"prediction: %d",(int)result);
                ////////////////////////////////////////////////////////
                error = abs((int)([tempdata getdesire:k]*5) - (int)result);
                switch (error) {
                    case 0:
                        zero ++;
                        break;
                    case 1:
                        one ++;
                        break;
                    case 2:
                        two ++;
                        break;
                    case 3:
                        three ++;
                        break;
                    case 4:
                        four ++;
                        break;

                    default:
                        break;
                }
                ////////////////////////////////////////////////////////
            }
        }
        NSLog(@"Zero: %d, One: %d, Two: %d, Three: %d, Four: %d",zero,one,two,three,four);

}
```

A2. Sequence diagrams



Sequence diagram for prediction

The sequence diagram for prediction shows the process of how the whole system works when system needs to make the prediction of songs level. When the timer says it is time to do the prediction operation, the music player controller will activate the hybrid engine. And then, the engine will do the prediction method.



Sequence diagram for user operation for rate a song

The sequence diagram for user operation for rate a song shows the interaction between user and the application. After user touch the screen to rate the song, the UI will receive the user's rating and the controller will process it and record it to database. And then, the controller will let UI to display the new rate for user.

A3. Result of Testing

Testing results from Run RAEnging

These results is generated when developer click the Run RAEngine button in the UI for developer. It shows the result of the hybrid engine. Firstly, it shows the result after core. The meaning of the result was shown in Subsection 4.2.2. And then, it shows the information about ANN like weight and threshold of each neuron before and after weight training process, and the number of iterations is also printed out.

```
2013-04-11 14:25:08.050 AIPlayer[11421:907] Max CP:50
2013-04-11 14:25:08.065 AIPlayer[11421:907] 0
2013-04-11 14:25:08.067 AIPlayer[11421:907] 0
2013-04-11 14:25:08.069 AIPlayer[11421:907] 1
2013-04-11 14:25:08.070 AIPlayer[11421:907] 1
2013-04-11 14:25:08.072 AIPlayer[11421:907] 1
```

```
2013-04-11 14:25:08.073 AIPlayer[11421:907] 0
2013-04-11 14:25:08.075 AIPlayer[11421:907] 1
2013-04-11 14:25:08.077 AIPlayer[11421:907] 0
2013-04-11 14:25:08.078 AIPlayer[11421:907] 0
2013-04-11 14:25:08.080 AIPlayer[11421:907] 1
2013-04-11 14:25:08.081 AIPlayer[11421:907] 1
2013-04-11 14:25:08.083 AIPlayer[11421:907] dp:0, low: 1, up : 34, apx: 0.029412
2013-04-11 14:25:08.085 AIPlayer[11421:907] dp:0, low: 0, up : 24, apx: 0.000000
2013-04-11 14:25:08.087 AIPlayer[11421:907] dp:0, low: 0, up : 30, apx: 0.000000
2013-04-11 14:25:08.089 AIPlayer[11421:907] dp:0, low: 0, up : 48, apx: 0.000000
2013-04-11 14:25:08.090 AIPlayer[11421:907] dp:0, low: 0, up : 40, apx: 0.000000
2013-04-11 14:25:08.092 AIPlayer[11421:907] dp:0, low: 0, up : 20, apx: 0.000000
2013-04-11 14:25:08.094 AIPlayer[11421:907]    (firstpicktimes = 5) ---------->(desirerate=
3.000000)
2013-04-11 14:25:08.096 AIPlayer[11421:907] dp:0, low: 3, up : 12, apx: 0.250000
2013-04-11 14:25:08.099 AIPlayer[11421:907] dp:0, low: 1, up : 16, apx: 0.062500
2013-04-11 14:25:08.101 AIPlayer[11421:907] dp:0, low: 1, up : 21, apx: 0.047619
2013-04-11 14:25:08.103 AIPlayer[11421:907] dp:0, low: 0, up : 24, apx: 0.000000
2013-04-11 14:25:08.105 AIPlayer[11421:907] dp:0, low: 5, up : 7, apx: 0.714286
2013-04-11 14:25:08.107 AIPlayer[11421:907]    (firstpicktimes = 1)    (ratesp = 2.000000)
---------->(desirerate= 3.000000)
2013-04-11 14:25:08.109 AIPlayer[11421:907]    (firstpicktimes = 2)    (ratesp = 3.000000)
---------->(desirerate= 3.000000)
2013-04-11 14:25:08.111 AIPlayer[11421:907]    (firstpicktimes = 4)    (ratesp = 3.000000)
---------->(desirerate= 3.000000)
2013-04-11 14:25:08.113 AIPlayer[11421:907]    (firstpicktimes = 4)    (ratesp = 2.000000)
---------->(desirerate= 3.000000)
2013-04-11 14:25:08.115 AIPlayer[11421:907] dp:0, low: 1, up : 3, apx: 0.333333
2013-04-11 14:25:08.117 AIPlayer[11421:907] dp:0, low: 0, up : 4, apx: 0.000000
2013-04-11 14:25:08.119 AIPlayer[11421:907] dp:0, low: 0, up : 4, apx: 0.000000
2013-04-11 14:25:08.121 AIPlayer[11421:907] dp:0, low: 0, up : 4, apx: 0.000000
2013-04-11 14:25:08.123 AIPlayer[11421:907]    (firstpicktimes = 2)    (maxscontinue = 3)
(ratesp = 2.000000) ---------->(desirerate= 3.000000)
2013-04-11 14:25:08.125 AIPlayer[11421:907] dp:0, low: 0, up : 2, apx: 0.000000
2013-04-11 14:25:08.128 AIPlayer[11421:907] dp:0, low: 0, up : 2, apx: 0.000000
2013-04-11 14:25:08.130 AIPlayer[11421:907] dp:0, low: 0, up : 2, apx: 0.000000
2013-04-11 14:25:08.132 AIPlayer[11421:907] dp:1, low: 0, up : 18, apx: 0.000000
2013-04-11 14:25:08.133 AIPlayer[11421:907] dp:1, low: 0, up : 23, apx: 0.000000
2013-04-11 14:25:08.135 AIPlayer[11421:907] dp:1, low: 0, up : 18, apx: 0.000000
2013-04-11 14:25:08.137 AIPlayer[11421:907] dp:1, low: 0, up : 17, apx: 0.000000
2013-04-11 14:25:08.139 AIPlayer[11421:907] dp:1, low: 0, up : 24, apx: 0.000000
```

```
2013-04-11 14:25:08.140 AIPlayer[11421:907] dp:1, low: 0, up : 17, apx: 0.000000
2013-04-11 14:25:08.142 AIPlayer[11421:907] dp:2, low: 0, up : 17, apx: 0.000000
2013-04-11 14:25:08.144 AIPlayer[11421:907] dp:2, low: 0, up : 18, apx: 0.000000
2013-04-11 14:25:08.146 AIPlayer[11421:907] dp:2, low: 0, up : 10, apx: 0.000000
2013-04-11 14:25:08.148 AIPlayer[11421:907] dp:2, low: 0, up : 23, apx: 0.000000
2013-04-11 14:25:08.150 AIPlayer[11421:907] dp:2, low: 0, up : 34, apx: 0.000000
2013-04-11 14:25:08.152 AIPlayer[11421:907] dp:2, low: 0, up : 13, apx: 0.000000
2013-04-11 14:25:08.154 AIPlayer[11421:907] dp:3, low: 1, up : 10, apx: 0.100000
2013-04-11 14:25:08.156 AIPlayer[11421:907] dp:3, low: 0, up : 21, apx: 0.000000
2013-04-11 14:25:08.157 AIPlayer[11421:907] dp:3, low: 0, up : 9, apx: 0.000000
2013-04-11 14:25:08.159 AIPlayer[11421:907] dp:3, low: 0, up : 22, apx: 0.000000
2013-04-11 14:25:08.161 AIPlayer[11421:907] dp:3, low: 0, up : 22, apx: 0.000000
2013-04-11 14:25:08.163 AIPlayer[11421:907] dp:3, low: 0, up : 9, apx: 0.000000
2013-04-11 14:25:08.165 AIPlayer[11421:907]    (firstpicktimes = 4) ---------->(desirerate=
5.000000)
2013-04-11 14:25:08.167 AIPlayer[11421:907] dp:3, low: 0, up : 4, apx: 0.000000
2013-04-11 14:25:08.169 AIPlayer[11421:907] dp:3, low: 2, up : 2, apx: 1.000000
2013-04-11 14:25:08.171 AIPlayer[11421:907] dp:3, low: 1, up : 5, apx: 0.200000
2013-04-11 14:25:08.173 AIPlayer[11421:907] dp:3, low: 0, up : 5, apx: 0.000000
2013-04-11 14:25:08.175 AIPlayer[11421:907] dp:3, low: 1, up : 4, apx: 0.250000
2013-04-11 14:25:08.177 AIPlayer[11421:907]    (firstpicktimes = 1)    (maxpcontinue = 1)
---------->(desirerate= 5.000000)
2013-04-11 14:25:08.179 AIPlayer[11421:907]    (firstpicktimes = 2)    (maxpcontinue = 1)
---------->(desirerate= 5.000000)
2013-04-11 14:25:08.182 AIPlayer[11421:907] dp:3, low: 0, up : 0, apx: 0.000000
2013-04-11 14:25:08.184 AIPlayer[11421:907] dp:3, low: 0, up : 0, apx: 0.000000
2013-04-11 14:25:08.186 AIPlayer[11421:907] dp:3, low: 0, up : 0, apx: 0.000000
2013-04-11 14:25:08.188 AIPlayer[11421:907] dp:3, low: 0, up : 0, apx: 0.000000
2013-04-11 14:25:08.190 AIPlayer[11421:907] dp:4, low: 0, up : 27, apx: 0.000000
2013-04-11 14:25:08.192 AIPlayer[11421:907] dp:4, low: 0, up : 11, apx: 0.000000
2013-04-11 14:25:08.194 AIPlayer[11421:907] dp:4, low: 0, up : 17, apx: 0.000000
2013-04-11 14:25:08.195 AIPlayer[11421:907] dp:4, low: 0, up : 35, apx: 0.000000
2013-04-11 14:25:08.197 AIPlayer[11421:907] dp:4, low: 0, up : 44, apx: 0.000000
2013-04-11 14:25:08.199 AIPlayer[11421:907] dp:4, low: 0, up : 19, apx: 0.000000
2013-04-11 14:25:08.201 AIPlayer[11421:907] dp:5, low: 0, up : 16, apx: 0.000000
2013-04-11 14:25:08.203 AIPlayer[11421:907] dp:5, low: 0, up : 10, apx: 0.000000
2013-04-11 14:25:08.205 AIPlayer[11421:907] dp:5, low: 0, up : 18, apx: 0.000000
2013-04-11 14:25:08.207 AIPlayer[11421:907] dp:5, low: 0, up : 33, apx: 0.000000
2013-04-11 14:25:08.209 AIPlayer[11421:907] dp:5, low: 0, up : 42, apx: 0.000000
2013-04-11 14:25:08.211 AIPlayer[11421:907] dp:5, low: 0, up : 17, apx: 0.000000
2013-04-11 14:25:08.213 AIPlayer[11421:907] dp:6, low: 0, up : 12, apx: 0.000000
```

2013-04-11 14:25:08.215 AIPlayer[11421:907] dp:6, low: 0, up : 9, apx: 0.000000
2013-04-11 14:25:08.217 AIPlayer[11421:907] dp:6, low: 0, up : 12, apx: 0.000000
2013-04-11 14:25:08.219 AIPlayer[11421:907] dp:6, low: 0, up : 21, apx: 0.000000
2013-04-11 14:25:08.220 AIPlayer[11421:907] dp:6, low: 0, up : 19, apx: 0.000000
2013-04-11 14:25:08.222 AIPlayer[11421:907] dp:6, low: 0, up : 15, apx: 0.000000
2013-04-11 14:25:08.224 AIPlayer[11421:907] dp:7, low: 0, up : 14, apx: 0.000000
2013-04-11 14:25:08.226 AIPlayer[11421:907] dp:7, low: 0, up : 15, apx: 0.000000
2013-04-11 14:25:08.228 AIPlayer[11421:907] dp:7, low: 0, up : 11, apx: 0.000000
2013-04-11 14:25:08.230 AIPlayer[11421:907] dp:7, low: 0, up : 16, apx: 0.000000
2013-04-11 14:25:08.232 AIPlayer[11421:907] dp:7, low: 0, up : 17, apx: 0.000000
2013-04-11 14:25:08.233 AIPlayer[11421:907] dp:7, low: 0, up : 6, apx: 0.000000
2013-04-11 14:25:08.236 AIPlayer[11421:907] data: 50
2013-04-11 14:25:08.244 AIPlayer[11421:907] hiddenlayer:
2013-04-11 14:25:08.246 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.029236....W3: -0.047782....W4: -0.086400....W5: 0.000000....W6: 0.157527....W7: 0.000000....W8: 0.000000....W9: 0.047127....W10: 0.112582....
2013-04-11 14:25:08.248 AIPlayer[11421:907] Threshold: -0.121091
2013-04-11 14:25:08.250 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.175636....W3: -0.049527....W4: -0.093600....W5: 0.000000....W6: -0.028145....W7: 0.000000....W8: 0.000000....W9: -0.204436....W10: -0.028582....
2013-04-11 14:25:08.252 AIPlayer[11421:907] Threshold: 0.054545
2013-04-11 14:25:08.253 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.054109....W3: 0.028364....W4: 0.004145....W5: 0.000000....W6: -0.110618....W7: 0.000000....W8: 0.000000....W9: -0.196364....W10: -0.025091....
2013-04-11 14:25:08.255 AIPlayer[11421:907] Threshold: -0.098182
2013-04-11 14:25:08.257 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: 0.112145....W3: 0.022473....W4: -0.078982....W5: 0.000000....W6: 0.151636....W7: 0.000000....W8: 0.000000....W9: -0.154691....W10: -0.138982....
2013-04-11 14:25:08.259 AIPlayer[11421:907] Threshold: -0.032073
2013-04-11 14:25:08.261 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.014182....W3: -0.008727....W4: -0.178255....W5: 0.000000....W6: 0.151855....W7: 0.000000....W8: 0.000000....W9: 0.068509....W10: -0.204000....
2013-04-11 14:25:08.263 AIPlayer[11421:907] Threshold: 0.145527
2013-04-11 14:25:08.265 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.020727....W3: -0.098836....W4: 0.051491....W5: 0.000000....W6: -0.033164....W7: 0.000000....W8: 0.000000....W9:

0.003709....W10: -0.180436....
2013-04-11 14:25:08.267 AIPlayer[11421:907] Threshold: 0.091855
2013-04-11 14:25:08.268 AIPlayer[11421:907]

outputLayer:
2013-04-11 14:25:08.270 AIPlayer[11421:907] The weights is:
W0: 0.004400....W1: -0.289200....W2: -0.170400....W3: -0.246400....W4: -0.311600....W5: -0.291600....
2013-04-11 14:25:08.272 AIPlayer[11421:907] Threshold: 0.192800

2013-04-11 14:25:10.254 AIPlayer[11421:907] iteration: 501
2013-04-11 14:25:10.256 AIPlayer[11421:907] error: 0.014820

2013-04-11 14:25:10.261 AIPlayer[11421:907] the statement fail: create table ann (id integer primary key autoincrement,ho int, numofan int, winfor text);
2013-04-11 14:25:10.348 AIPlayer[11421:907] hiddenlayer:
2013-04-11 14:25:10.349 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.132622....W3: 0.403171....W4: -0.854927....W5: 0.000000....W6: -0.370534....W7: 0.000000....W8: 0.000000....W9: -0.330775....W10: 1.228717....
2013-04-11 14:25:10.351 AIPlayer[11421:907] Threshold: 0.316882
2013-04-11 14:25:10.353 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.128909....W3: 0.042159....W4: -0.099216....W5: 0.000000....W6: -0.017182....W7: 0.000000....W8: 0.000000....W9: -0.187168....W10: -0.084990....
2013-04-11 14:25:10.354 AIPlayer[11421:907] Threshold: 0.022036
2013-04-11 14:25:10.356 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: -0.073752....W3: 0.149793....W4: -0.033260....W5: 0.000000....W6: -0.151379....W7: 0.000000....W8: 0.000000....W9: -0.200983....W10: -0.135155....
2013-04-11 14:25:10.357 AIPlayer[11421:907] Threshold: -0.126060
2013-04-11 14:25:10.359 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: 0.131325....W3: 0.425491....W4: -0.373079....W5: 0.000000....W6: -0.041000....W7: 0.000000....W8: 0.000000....W9: -0.227027....W10: -0.487784....
2013-04-11 14:25:10.361 AIPlayer[11421:907] Threshold: -0.090822
2013-04-11 14:25:10.363 AIPlayer[11421:907] The weights is:
W0: 0.000000....W1: 0.000000....W2: 0.187677....W3: 0.411321....W4: -0.436447....W5: 0.000000....W6: -0.061820....W7: 0.000000....W8: 0.000000....W9: -0.001505....W10: -0.619532....
2013-04-11 14:25:10.365 AIPlayer[11421:907] Threshold: 0.061318

```
2013-04-11 14:25:10.367 AIPlayer[11421:907] The weights is:
W0:  0.000000....W1:  0.000000....W2:  0.236929....W3:  0.038265....W4:
0.078397....W5:  0.000000....W6:  -0.090801....W7:  0.000000....W8:  0.000000....W9:
0.277100....W10:  -0.828647....
2013-04-11 14:25:10.369 AIPlayer[11421:907] Threshold: -0.055775
2013-04-11 14:25:10.370 AIPlayer[11421:907]

outputLayer:
2013-04-11 14:25:10.372 AIPlayer[11421:907] The weights is:
W0:  1.637192....W1:  -0.411229....W2:  -0.361497....W3:  -0.717116....W4:
-0.813308....W5:  -0.669997....
2013-04-11 14:25:10.374 AIPlayer[11421:907] Threshold: -0.757762
```

Testing results from Run withoutCA

These results is generated when developer click the Run withoutCA button in the UI for developer. It will show the result that only ANN engine runs. It shows the information about ANN like weight and threshold of each neuron before and after weight training process, and the number of iterations is also printed out.

```
2013-04-11 14:35:07.045 AIPlayer[11421:907] 1
2013-04-11 14:35:07.046 AIPlayer[11421:907] the statement fail: create table dsong
(songindex integer primary key autoincrement, songname text, artist text, rateartist double,
album text, ratealbum double, genre text, rategenre double, skiptime int, playtime int,
pcontinue int, scontinue int, maxpcontinue int, maxscontinue int, firstpicktimes int, ratesp
double,desirerate double, userrate double,predictrate double);
2013-04-11 14:35:09.018 AIPlayer[11421:907] hiddenlayer:
2013-04-11 14:35:09.020 AIPlayer[11421:907] The weights is:
W0:  -0.072000....W1:  0.154255....W2:  0.108655....W3:  -0.154473....W4:
-0.172145....W5:  0.100145....W6:  0.043855....W7:  0.017018....W8:  -0.182182....W9:
0.133309....W10:  -0.010036....
2013-04-11 14:35:09.022 AIPlayer[11421:907] Threshold: -0.013309
2013-04-11 14:35:09.024 AIPlayer[11421:907] The weights is:
W0:  0.197236....W1:  0.017236....W2:  -0.210109....W3:  -0.111055....W4:
0.077891....W5:  -0.034255....W6:  0.087055....W7:  -0.184145....W8:  -0.054764....W9:
0.060218....W10:  -0.049745....
2013-04-11 14:35:09.026 AIPlayer[11421:907] Threshold: 0.098182
2013-04-11 14:35:09.028 AIPlayer[11421:907] The weights is:
```

W0: 0.160800....W1: -0.035345....W2: 0.200291....W3: 0.000436....W4: 0.141818....W5: 0.187200....W6: -0.055636....W7: 0.016582....W8: 0.217527....W9: 0.198982....W10: 0.093818....
2013-04-11 14:35:09.030 AIPlayer[11421:907] Threshold: -0.207491
2013-04-11 14:35:09.032 AIPlayer[11421:907] The weights is:
W0: 0.110618....W1: 0.055636....W2: 0.203564....W3: 0.035345....W4: 0.214255....W5: -0.202036....W6: -0.181309....W7: 0.144436....W8: 0.003709....W9: -0.068291....W10: -0.155564....
2013-04-11 14:35:09.033 AIPlayer[11421:907] Threshold: 0.046909
2013-04-11 14:35:09.035 AIPlayer[11421:907] The weights is:
W0: -0.116073....W1: -0.026400....W2: -0.128073....W3: 0.084655....W4: 0.050618....W5: 0.094255....W6: 0.211200....W7: -0.022691....W8: -0.040582....W9: -0.016364....W10: -0.181091....
2013-04-11 14:35:09.037 AIPlayer[11421:907] Threshold: 0.106473
2013-04-11 14:35:09.039 AIPlayer[11421:907] The weights is:
W0: 0.047564....W1: -0.108873....W2: 0.032509....W3: -0.158182....W4: 0.140727....W5: -0.177818....W6: 0.006982....W7: 0.057818....W8: 0.114545....W9: -0.176073....W10: 0.003273....
2013-04-11 14:35:09.041 AIPlayer[11421:907] Threshold: 0.057818
2013-04-11 14:35:09.043 AIPlayer[11421:907]

outputLayer:
2013-04-11 14:35:09.045 AIPlayer[11421:907] The weights is:
W0: -0.373200....W1: 0.037600....W2: 0.026800....W3: -0.163200....W4: 0.374000....W5: -0.323200....
2013-04-11 14:35:09.047 AIPlayer[11421:907] Threshold: 0.190400

2013-04-11 14:35:10.998 AIPlayer[11421:907] iteration: 501
2013-04-11 14:35:11.000 AIPlayer[11421:907] error: 0.005949

2013-04-11 14:35:11.005 AIPlayer[11421:907] the statement fail: create table ann (id integer primary key autoincrement,ho int, numofan int, winfor text);
2013-04-11 14:35:11.091 AIPlayer[11421:907] hiddenlayer:
2013-04-11 14:35:11.093 AIPlayer[11421:907] The weights is:
W0: -0.631337....W1: 0.466718....W2: -0.001112....W3: 0.000324....W4: 0.154525....W5: 0.287026....W6: 0.779414....W7: -0.150615....W8: 0.213172....W9: 0.023191....W10: -0.944913....
2013-04-11 14:35:11.095 AIPlayer[11421:907] Threshold: -0.088702
2013-04-11 14:35:11.097 AIPlayer[11421:907] The weights is:
W0: 0.234846....W1: -0.142224....W2: -0.053418....W3: -0.331021....W4: 0.664770....W5: -0.480510....W6: -0.157454....W7: 0.210733....W8: -0.425304....W9:

```
0.313773....W10:    0.777466....
2013-04-11 14:35:11.098 AIPlayer[11421:907] Threshold: 0.086632
2013-04-11 14:35:11.100 AIPlayer[11421:907] The weights is:
W0:    0.049895....W1:    -0.138043....W2:    0.111198....W3:    -0.288357....W4:
0.200294....W5:    0.032969....W6:    0.014056....W7:    0.054312....W8:    0.082332....W9:
0.189557....W10:    0.087565....
2013-04-11 14:35:11.101 AIPlayer[11421:907] Threshold: -0.200292
2013-04-11 14:35:11.103 AIPlayer[11421:907] The weights is:
W0:    -0.199463....W1:    0.221180....W2:    0.161687....W3:    0.037485....W4:
0.476501....W5:    0.096618....W6:    0.205604....W7:    0.038005....W8:    0.194636....W9:
-0.063828....W10:    -0.654981....
2013-04-11 14:35:11.105 AIPlayer[11421:907] Threshold: 0.002148
2013-04-11 14:35:11.107 AIPlayer[11421:907] The weights is:
W0:    0.086338....W1:    0.034965....W2:    0.004065....W3:    0.106990....W4:
0.126517....W5:    0.213344....W6:    0.349866....W7:    0.161436....W8:    0.052734....W9:
0.121827....W10:    0.071538....
2013-04-11 14:35:11.109 AIPlayer[11421:907] Threshold: 0.065560
2013-04-11 14:35:11.111 AIPlayer[11421:907] The weights is:
W0:    -0.548073....W1:    0.285923....W2:    -0.037998....W3:    -0.278002....W4:
0.495083....W5:    0.390254....W6:    0.438962....W7:    -0.038314....W8:    0.460675....W9:
-0.278350....W10:    -0.887408....
2013-04-11 14:35:11.113 AIPlayer[11421:907] Threshold: -0.009058
2013-04-11 14:35:11.115 AIPlayer[11421:907]

outputLayer:
2013-04-11 14:35:11.117 AIPlayer[11421:907] The weights is:
W0:    -1.006299....W1:    0.975049....W2:    0.600190....W3:    -0.495135....W4:
0.807469....W5:    -1.010940....
2013-04-11 14:35:11.119 AIPlayer[11421:907] Threshold: -0.363504
```

Testing results of Prediction

These results is generated when developer click the Prediction button in the UI for developer. It will show the result of using the hybrid engine to make the prediction. In the result, the desire rate and the predict rate of a song are showing. And the conclusion about the error between desire rate and the predict rate.

```
2013-04-11 14:37:59.582 AIPlayer[11421:907] 1
```

2013-04-11 14:37:59.583 AIPlayer[11421:907] the statement fail: create table dsong (songindex integer primary key autoincrement, songname text, artist text, rateartist double, album text, ratealbum double, genre text, rategenre double, skiptime int, playtime int, pcontinue int, scontinue int, maxpcontinue int, maxscontinue int, firstpicktimes int, ratesp double,desirerate double, userrate double,predictrate double);
2013-04-11 14:38:01.557 AIPlayer[11421:907] songname:04 You and I.m4a
2013-04-11 14:38:01.568 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.569 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.571 AIPlayer[11421:907] songname:Avril Lavigne - How You Remind Me.mp3
2013-04-11 14:38:01.587 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.589 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:01.590 AIPlayer[11421:907] songname:Avril Lavigne - Innocence.mp3
2013-04-11 14:38:01.603 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.604 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.606 AIPlayer[11421:907] songname:Avril Lavigne - Wish You Were Here.mp3
2013-04-11 14:38:01.618 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:01.620 AIPlayer[11421:907] prediction: 5
2013-04-11 14:38:01.622 AIPlayer[11421:907] songname:Ben E. King - Stand By Me.mp3
2013-04-11 14:38:01.632 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.633 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:01.635 AIPlayer[11421:907] songname:Beyonce - Halo.mp3
2013-04-11 14:38:01.651 AIPlayer[11421:907] desire: 5
2013-04-11 14:38:01.653 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:01.654 AIPlayer[11421:907] songname:Beyond、黄家驹 - 不再犹豫.mp3
2013-04-11 14:38:01.669 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.670 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:01.671 AIPlayer[11421:907] songname:Daniel - Free Loop.mp3
2013-04-11 14:38:01.684 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.686 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:01.687 AIPlayer[11421:907] songname:Green Day - The Forgotten.mp3
2013-04-11 14:38:01.696 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.698 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.700 AIPlayer[11421:907] songname:Jack Johnson - Better Together.mp3
2013-04-11 14:38:01.714 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.716 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.717 AIPlayer[11421:907] songname:Jessie J - Price Tag.mp3
2013-04-11 14:38:01.731 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.733 AIPlayer[11421:907] prediction: 2

2013-04-11 14:38:01.735 AIPlayer[11421:907] songname:Jessie J - We Found Love (Rihanna Cover).mp3
2013-04-11 14:38:01.748 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.750 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.752 AIPlayer[11421:907] songname:Louis Armstrong - What A Wonderful World.mp3
2013-04-11 14:38:01.762 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.763 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:01.766 AIPlayer[11421:907] songname:Lucky - Jason Mraz&colbie Caillat.mp3
2013-04-11 14:38:01.780 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:01.781 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:01.783 AIPlayer[11421:907] songname:Maddi Jane - Price Tag.mp3
2013-04-11 14:38:01.796 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.797 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:01.799 AIPlayer[11421:907] songname:Natasha Thomas - It's Over Now - 鳄鱼广告歌.mp3
2013-04-11 14:38:01.812 AIPlayer[11421:907] desire: 5
2013-04-11 14:38:01.814 AIPlayer[11421:907] prediction: 5
2013-04-11 14:38:01.816 AIPlayer[11421:907] songname:Natasha Thomas - Let Me Show You The Way.mp3
2013-04-11 14:38:01.825 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:01.827 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:01.829 AIPlayer[11421:907] songname:Natasha Thomas、p!nk - Song.mp3
2013-04-11 14:38:01.844 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:01.846 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:01.847 AIPlayer[11421:907] songname:Olly Murs、Flo Rida - Troublemaker.mp3
2013-04-11 14:38:01.860 AIPlayer[11421:907] desire: 1
2013-04-11 14:38:01.862 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:01.863 AIPlayer[11421:907] songname:Psy - 江南style.mp3
2013-04-11 14:38:01.875 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.877 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.879 AIPlayer[11421:907] songname:S.B.D.W - 世界末日.mp3
2013-04-11 14:38:01.888 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.890 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.892 AIPlayer[11421:907] songname:S.B.D.W - 我承认 - 很好听一首追女孩子的歌 R&b.mp3
2013-04-11 14:38:01.908 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.909 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.911 AIPlayer[11421:907] songname:S.B.D.W - 起点 - 咻比嘟哗.mp3

2013-04-11 14:38:01.925 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:01.927 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:01.928 AIPlayer[11421:907] songname:S.B.D.W - 野孩子.mp3
2013-04-11 14:38:01.940 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.942 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.944 AIPlayer[11421:907] songname:Sam Tsui、2010 - If I Die Young.mp3
2013-04-11 14:38:01.953 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:01.954 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:01.956 AIPlayer[11421:907] songname:Tank - 非你莫属.mp3
2013-04-11 14:38:01.970 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:01.972 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.973 AIPlayer[11421:907] songname:Taozhe - 寂寞的季节.mp3
2013-04-11 14:38:01.988 AIPlayer[11421:907] desire: 1
2013-04-11 14:38:01.989 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:01.991 AIPlayer[11421:907] songname:The Voice - If I Ain't Got You.mp3
2013-04-11 14:38:02.005 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:02.007 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:02.008 AIPlayer[11421:907] songname:What are words .mp3
2013-04-11 14:38:02.017 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.019 AIPlayer[11421:907] prediction: 5
2013-04-11 14:38:02.021 AIPlayer[11421:907] songname:³ÂÞÈÑ¸ - È«ÊÀ½çÊ§ÃßB.mp3
2013-04-11 14:38:02.035 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:02.036 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:02.038 AIPlayer[11421:907] songname:世界各地街头艺人 Stand By Me(1).mp3
2013-04-11 14:38:02.051 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:02.052 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:02.054 AIPlayer[11421:907] songname:周杰伦 - 乌克丽丽.mp3
2013-04-11 14:38:02.068 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.070 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:02.072 AIPlayer[11421:907] songname:周杰伦 - 傻笑.mp3
2013-04-11 14:38:02.081 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:02.082 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:02.084 AIPlayer[11421:907] songname:周杰伦 - 公公偏头痛.mp3
2013-04-11 14:38:02.100 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:02.101 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:02.103 AIPlayer[11421:907] songname:周杰伦 - 哪里都是你.mp3
2013-04-11 14:38:02.116 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.118 AIPlayer[11421:907] prediction: 4

2013-04-11 14:38:02.119 AIPlayer[11421:907] songname:周杰伦 - 大笨钟.mp3
2013-04-11 14:38:02.132 AIPlayer[11421:907] desire: 1
2013-04-11 14:38:02.134 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:02.135 AIPlayer[11421:907] songname:周杰伦 - 手语.mp3
2013-04-11 14:38:02.144 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:02.146 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:02.148 AIPlayer[11421:907] songname:周杰伦 - 梦想启动.mp3
2013-04-11 14:38:02.164 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:02.165 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:02.167 AIPlayer[11421:907] songname:周杰伦 - 爱你没差.mp3
2013-04-11 14:38:02.181 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.182 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:02.184 AIPlayer[11421:907] songname:孙燕姿 - 我不难过.mp3
2013-04-11 14:38:02.196 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.198 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:02.199 AIPlayer[11421:907] songname:孙燕姿 - 我怀念的.mp3
2013-04-11 14:38:02.209 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.210 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:02.212 AIPlayer[11421:907] songname:常石磊 - I Still Believe.mp3
2013-04-11 14:38:02.226 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:02.227 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:02.229 AIPlayer[11421:907] songname:曲婉婷 - 我的歌声里.mp3
2013-04-11 14:38:02.243 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:02.245 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:02.246 AIPlayer[11421:907] songname:梁心颐、周杰伦 - 比较大的大提琴.mp3
2013-04-11 14:38:02.260 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:02.262 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:02.264 AIPlayer[11421:907] songname:玛利亚凯丽 - I Still Believe.mp3
2013-04-11 14:38:02.274 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:02.275 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:02.277 AIPlayer[11421:907] songname:陈奕迅 - 一丝不挂.mp3
2013-04-11 14:38:02.292 AIPlayer[11421:907] desire: 3
2013-04-11 14:38:02.293 AIPlayer[11421:907] prediction: 3
2013-04-11 14:38:02.295 AIPlayer[11421:907] songname:陈奕迅 - 不如不见.mp3
2013-04-11 14:38:02.308 AIPlayer[11421:907] desire: 2
2013-04-11 14:38:02.310 AIPlayer[11421:907] prediction: 2
2013-04-11 14:38:02.311 AIPlayer[11421:907] songname:陈奕迅 - 兄妹.mp3
2013-04-11 14:38:02.325 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.327 AIPlayer[11421:907] prediction: 4
2013-04-11 14:38:02.329 AIPlayer[11421:907] songname:陈奕迅 - 淘汰.mp3

2013-04-11 14:38:02.338 AIPlayer[11421:907] desire: 4
2013-04-11 14:38:02.340 AIPlayer[11421:907] prediction: 5
2013-04-11 14:38:02.342 AIPlayer[11421:907] songname:陈奕迅、王菲 - 因为爱情.mp3
2013-04-11 14:38:02.358 AIPlayer[11421:907] desire: 5
2013-04-11 14:38:02.359 AIPlayer[11421:907] prediction: 5

2013-04-11 14:38:02.361 AIPlayer[11421:907] Zero: 29, One: 19, Two: 2, Three: 0, Four: 0

2013-04-11 14:38:02.362 AIPlayer[11421:907] hiddenlayer:
2013-04-11 14:38:02.364 AIPlayer[11421:907] The weights is:
W0: -0.631337....W1: 0.466718....W2: -0.001112....W3: 0.000324....W4: 0.154525....W5: 0.287026....W6: 0.779414....W7: -0.150615....W8: 0.213172....W9: 0.023191....W10: -0.944913....
2013-04-11 14:38:02.365 AIPlayer[11421:907] Threshold: -0.088702
2013-04-11 14:38:02.367 AIPlayer[11421:907] The weights is:
W0: 0.234846....W1: -0.142224....W2: -0.053418....W3: -0.331021....W4: 0.664770....W5: -0.480510....W6: -0.157454....W7: 0.210733....W8: -0.425304....W9: 0.313773....W10: 0.777466....
2013-04-11 14:38:02.369 AIPlayer[11421:907] Threshold: 0.086632
2013-04-11 14:38:02.371 AIPlayer[11421:907] The weights is:
W0: 0.049895....W1: -0.138043....W2: 0.111198....W3: -0.288357....W4: 0.200294....W5: 0.032969....W6: 0.014056....W7: 0.054312....W8: 0.082332....W9: 0.189557....W10: 0.087565....
2013-04-11 14:38:02.373 AIPlayer[11421:907] Threshold: -0.200292
2013-04-11 14:38:02.375 AIPlayer[11421:907] The weights is:
W0: -0.199463....W1: 0.221180....W2: 0.161687....W3: 0.037485....W4: 0.476501....W5: 0.096618....W6: 0.205604....W7: 0.038005....W8: 0.194636....W9: -0.063828....W10: -0.654981....
2013-04-11 14:38:02.377 AIPlayer[11421:907] Threshold: 0.002148
2013-04-11 14:38:02.379 AIPlayer[11421:907] The weights is:
W0: 0.086338....W1: 0.034965....W2: 0.004065....W3: 0.106990....W4: 0.126517....W5: 0.213344....W6: 0.349866....W7: 0.161436....W8: 0.052734....W9: 0.121827....W10: 0.071538....
2013-04-11 14:38:02.381 AIPlayer[11421:907] Threshold: 0.065560
2013-04-11 14:38:02.383 AIPlayer[11421:907] The weights is:
W0: -0.548073....W1: 0.285923....W2: -0.037998....W3: -0.278002....W4: 0.495083....W5: 0.390254....W6: 0.438962....W7: -0.038314....W8: 0.460675....W9: -0.278350....W10: -0.887408....
2013-04-11 14:38:02.385 AIPlayer[11421:907] Threshold: -0.009058
2013-04-11 14:38:02.387 AIPlayer[11421:907]

```
outputLayer:
2013-04-11 14:38:02.389 AIPlayer[11421:907] The weights is:
W0:  -1.006299....W1:  0.975049....W2:  0.600190....W3:  -0.495135....W4:
0.807469....W5:  -1.010940....
2013-04-11 14:38:02.390 AIPlayer[11421:907] Threshold: -0.363504
```