MusicSense - Smart player in android environment

Amit Dvir[1], Sefi Erlich[1], Aviv Levanon[1], Maria Vinogradov[2], Idan Nahmias[2] and Chen Maman[2]

[1] *Center for Cyber Technologies, Department of Computer Science, Ariel University, Israel*
[2] *Department of Computer Science, Ariel University, Israel*

Abstract

Today, when smartwatches are common devices, it provides the user to adjust personal preferences using the advanced technologies in this area.
One of those technologies is a music player, built into the phone, while allowing control over songs in the watch itself.
Today, current solutions depend on the user - he tells the music player what songs he likes, or on other users who liked songs and those songs will be played to the user.Those solutions don't take the personal status and condition in hand.
Therefore, the idea of this project shows possibility to improve the technologies that already exist in the watch, such improvement will allow the music player to choose the right song to certain moment and saving the user time to find the song by himself.

Introduction

The project we are working on will show a solution to the following problems: The first part to solve will be the way we collect, save and contain the data. This data will be saved on a CSV files that will manage all the information. The log files will contain Vectors representing different information about sensors and user activity. The next challenge will be to combine the music player and the sensor's data to a format that can be used by machine learning.
Finally, last mission will be handling the communication with the machine learning. It is important to make sure we have similar language for the data we send to and receive from the machine model. The result action chosen from the data sent must be part of the possible outputs for our model, allowing the music player to understand which action the machine choose for the current situation.
In advance, on a further stage, we will need to provide a solution for errors that will occur during the implementation part. With this information in mind, the application will need to have the ability to learn from its mistakes. Therefor, the learning-implementing process will always have to work simultaneously.
This learning as we mention, in further stage of the project, will need to be collected from the user interaction with the player, or feed backs for songs during playtime. The ability to learn during the implementation will allow us to have a powerful-more capable machine that will predict better how to personalize each song for the user as its interacts with him.

Android sensor

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful if you want to monitor three-dimensional device movement or positioning. For example, a game might track readings from a device's gravity sensor to infer complex user gestures and motions, including tilt, shake, rotation, or swing. Likewise, a weather application might use a device's temperature sensor and humidity sensor to calculate and report the dew point, or a travel application might use the geomagnetic field sensor and accelerometer to report a compass bearing.

- Motion sensors
  These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

- Environmental sensors
  These sensors measure various environmental parameters, including ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

- Position sensors
  These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

Music is a big part for the most of us. We like listening to music during our daily activity, whether while we workout, jog, travel, and even when we sleep. Using all the available sensors today's technology provide, allow us to monitor the activity the user is making, and match the type of music for that activity. For example, when i am jogging outside,

my heart rate is high in average, my steps counts increase fast, my location changes in medium rate, my arms are moving in a specific direction and so on. All these facts can be recorded in a smartphone using the correct sensors.[4]

Sensors and music player

Music is a big part for the most of us. We like listening to music during our daily activity, whether while we workout, jog, travel, and even when we sleep. Using all the available sensors today's technology provide, allow us to monitor the activity the user is making, and match the type of music for that activity. For example, when i am jogging outside, my heart rate is high in average, my steps counts increase fast, my location changes in medium rate, my arms are moving in a specific direction and so on. All these facts can be recorded in a smartphone using the correct sensors.

Project implementation

To implement this project, we have built an application for android smartphones.
This application main activity is a music player with the following options to the user: Play, stop, next, previous, forward, backwards, shuffle, repeat, and prediction.
Also, the user can navigate to the song list, as well as to a screen that shows information about the location of the device.
The application also runs on the smartwatch and the music player can be controlled directly from it. The application displays the data about the smartwatch sensors while the music is playing. The collected data from the sensors is raw, and is saved on the smartphone as CSV files. While the user is using the application, an activity file stores all the actions the user made in the application. Once the application restarts, an automated operation takes place: We iterate over activity file from the time the application started until the current time. Each line in Activity.csv lets us know all the information about the song played, the time it occurred and the action that happened after that. Using this information, we can access the raw data stored for each sensor, and save it in a vector of vectors of double values (Vector<Vector<Double>>).
The vector is being processed and an arithmetic functions are applied to all vectors, finally returning a processed vector representing the concrete data from a specific action (Vector<Double>).
The concrete data is either stored in DataVector.csv file to be used as training data set for machine learning, or to create new instances for the machine to learn/predict.

Machine learning over android environment

Use of machine learning requires much more computational power than normal tasks. Therefore, the CPU or GPU that needs to deal with those kinds of processing power must be capable for the task. Usually, the smartphone's cpu will not deal with the calculations needed in the training part(will be discussed in the next section). The process of implementing a machine learning goes as follows:

- Choosing the type of output (depends on the problem we wish to solve).
- Create the data set for training and testing.
- Choosing the algorithm that fits our needs.
- Produce A model that can make prediction.
- Update the model with new data over time to make it more efficient.

Choosing the type of output: Regression VS Classification

Is it a bird? Is it a plane? Is it 3.14159?
When we think of AI we think of an algorithm that can make computer actions that require human intelligence to complete. In machine learning that action can be decided by 2 ways: regression and classification. Regression analysis - In statistics, a mathematical method of modeling the relationships among three or more variables. It is used to predict the value of one variable given the values of the others. For example, a model might estimate sales based on age and gender.
Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Examples are assigning a given email to the "spam" or "non-spam" class, and assigning a diagnosis to a given patient based on observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition.

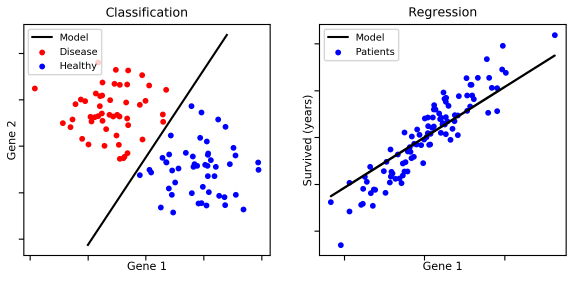

Figure 1: Regression VS Classification example

Create the data set for training and testing

By now, you should figure that our problem is a classification problem and therefor, the need for training data is mandatory. The first thing to consider for classification is our class group(e.g spam, not spam). This group will contain the possible outcomes from our machine, and this group is final. After that we can talk about attributes and instances: Attributes - the category for a single value. Instance - vector of attributes values and a class value (the class is also an attribute).
With this, we can create instances that will represent our training set and/or our testing set, this set should be as big as possible and as correct as possible if we want good predictions in the future.

Choosing the algorithm that fits our needs

There are quite a few machine learning algorithms and the big question is which should you use? Well, the answer is depends. Depends on the problem you are trying to solve, the type of data you provide, its size , quality and nature. Trying and testing them will help you decide the best.

Produce A model that can make prediction, and update it over time

Now that we have a data set for training, and an algorithm for classification, a model can be created. This model is the result of the classification of the training set. Now it is created, we can provide new instance and use the model to predict outcomes. Before we start using this model, we want to test our algorithm and see how good this model of ours. The testing set is just for that, validating and confirming that our machine actually work. Creating the model is good and it gives the machine the sense of what output it should provide, but it's not enough. Our goal is to create a machine that evolve and become precise as much as possible. Therefor , after creating the model it is important to keep it updated. We can think of it as adding more instances to the training set.

Machine learning in MusicSense

To implement machine learning in our project, we used Android studio[4], Intellij[5], Weka for java[6], and openCSV[7]. As described before, the machine type is classification. Therefor, some preparation was needed. For the training set, we had to collect data from the smartphone and the watch's sensors.To understand which action the user made with specific set of sensors, every action made in time t_i was saved in CSV file stating: the action made, the time and date, the song title, the song's progress state, and the previous song played. This help us to get access to the data collected from the sensors. For example, after an action q some action p occurs, with the information collected, we can collect all data from all sensors that happen from time t_q to time t_p.
We save and keep this data as our training set, which means that amounts of instances in the training set will be the same as amounts of actions the user made before the extraction(not doing an action can be considered action. For example: not doing any action while a song is playing can point to the fact that the user likes the song). For the algorithm, we wanted something simple and fast. Our instances were in the size of 44 values each, so kNN (k-nearest-neighbors) with k=2 was the algorithm of our choice. At this point comes the complication that we spoke of at the beginning of this section:
The fact that computing the model cannot occur inside the android environment. The creation of the model was made outside of android. The training data is sent to a computer. Using java and Weka, this computer create the model with the training data and with the algorithm chosen. The model file is saved back on the smartphone. The fact that we had to send the data away from our application raises the question of how do the user know when to send it and to whom?
The answer is that the user don't need to send anything. The application is being installed with the model file already. We create a generic model that can contain all the actions available to the user and use it as our class attribute in the training set. From here, the learning part starts, and each action the user make, is being learned by the model, saving it for better predictions. Eventually, the application will be smart enough to predict the user preferences only by measuring his physical state from the sensors.

Related works

Human activity and emotion aware mobile music player[1] presents a device that aims to integrate awareness of human activity and musical preferences to produce an adaptive system that plays the contextually correct music. The XPod project introduces a "smart" music player that learns its user's preferences and activity, and tailors its music selections accordingly.
The XPod project used an special device that the user must wear so it could be able to collect the data needed. Groove: Smart Music Player[2] application will provide new songs and reacts to the user playlist. Sony Smart B-Trainer[3] is an all-in-one device that provides the earphones with the application inside. This software take measurements from the heart rate monitor the user is using and suggest music to the user based on the data.
This software main activity is to help trainers while its not only a music player, but also Advisor. While

this product uses the sensors from the user, it cannot connect to every phone, and it forces the user to use the earphone provided.

Conclusions

MusicSense have great potential, it works with android and wear OS, there are countless amounts of android phones and currently 41 smartwatches not including 12 future watches to be released soon. This gives much more platforms for MusicSense to work on, allowing many users to enjoy smart music selection.

References

[1] XPod a human activity and emotion aware mobile music player.In Proceedings of the International Conference on Mobile Technology, Applications and Systems, November 2005.
Sandor Dornbush, Kevin Fisher, Kyle McKay, Alex Prikhodko, and Zary Segall

[2] Groove: Smart Music Player.

[3] Sony Smart B-Trainer

[4] Android sensors overview:
texttthttps://developer.android.com/

[5] Intellij
`https://www.jetbrains.com/idea/`

[6] Weka for android
`https://github.com/rjmarsan/Weka-for-Android`

[7] openCSV
`http://opencsv.sourceforge.net/`

[8] Figure 1
`https://medium.freecodecamp.org/using-machine-learning-to-predic-the-quality-of-wines-9e2e13d7480d`