# A Neuro and Fuzzy Approach to Music Classification

Ashley Holmes

December 16, 2014

# Contents

**Abstract**

The accessibility and personalization of music is widely important in today's world. Nearly everybody has a smartphone, enabling them to stream music wherever they go. There are countless programs dedicated to personalizing your music tastes, such as Pandora, Apple's Genius Radio, and Spotify.

# 1 List of Figures

# 2 Introduction

The accessibility and personalization of music is widely important in today's world. Nearly everybody has a smartphone, enabling them to stream music wherever they go. There are countless programs dedicated to personalizing your music tastes, such as Pandora, Apple's Genius, Spotify, 8tracks, Last.fm, Grooveshark, iHeartRadio, and countless others which are less popular. Making the perfect playlist is an envied skill in this pervasive world of music. As a systems scientist that has always had a profound interest in music, I have often wondered just how Pandora, etc. have created (and perfected) their system of creating playlists. Concluding that the essential part of any playlist is having songs with similar speeds, I decided to apply the soft computing concepts of a fuzzy inference system to measure the perceived speeds of different songs, and an artificial neural network to train a machine to learn to group songs of similar speeds together.

First, I'll explain the relatively brief history of music classification, as well as the most successful companies in the field and their specific approaches. There I'll also discuss a bit about the context of the application of this project and why it is a challenging problem. Then, I'll explain the background on the approach that I took for this project, including some background on artificial neural networks and fuzzy inference systems. Next, I'll go over the details of the implementation of my project, including how I collected data, transformed the data using a Mamdani-style fuzzy inference system, and then trained a neural network with the data. Finally, I'll discuss the results of this project and some conclusions that I came to. Lastly, I'll end with how successful the problem was compared to my intended goals, discuss what I would do differently next time, and consider future endevaors to extend or expand on this project.

# 3 Background on the Application

Pandora is a well-known music program intended to create playlists of similar-sounding songs based on a number of factors. The wide success of this program has spawned the creation of other music-organizing programs such as Spotify, Grooveshark, Last.fm, 8tracks, iHeartRadio, and Apple's Genius Radio. These programs have revolutionized the organization and classification of music. Although Spotify and Apple Radio have grown and become much better, many users still swear by Pandora for its unique and perfected method for selection of songs. Amadou Diallo wrote in Forbes' magazine, "Pandora's matches show close attention paid to musical styles, instrumentation and emotional content or mood. I?d feel confident using a Pandora station as a virtual DJ at a party, for example without worrying about it tossing in an embarrassing clunker that required an immediate thumbs down." [12]

Pandora's main algorithm, called the Music Genome Project, analyzes up to 450 distinct musical characteristics, including but not limited to: melody, harmony, instrumentation, rhythm, vocals, and lyrics. [6] Grooveshark describes itself as an "ecosystem." Last.fm uses a collaborative filtering algorithm [7], Apple claims to have its own "clever algorithm," (with little to no detail on what it actually does), and Spotify works by using its incredibly large library, sometimes even adding albums the day they are released. [1]

Since Pandora was the first (and today is still the most successful) music program that learns from your preferences, I will focus on the history of this program. Tim Westergren, the founder, was a musician who eventually found himself composing scores for films. After playing in various bands with varying degrees of success, he found this job

to be very different than the his previous engagement in said bands. Instead of freestyling jams with his bandmates, Westergren felt that there was more precision, more calculation to this job. [22] After working with several directors, he noticed that different directors preferred different styles of musical composition. Anyone that has ever played for more than one musical director, whether in high school, or professional, can attest to this. As a result of this conclusion, Westergren was curious as to whether he could "codify" the specific tastes of each of the directors that he worked for. [10] By "codifying" tastes, he meant that he intended to break them down into something predictable, mathematical, or ideally, formulaic.

Westergren began to look at music in a very analytical way, which led him to start a company that focused on decoding music. His idea tranformed into personalization of music, with the intention of introducing people to the type music that they liked based on various factors, including their listening history. He did this by creating the aforementioned Music Genome Project, whose goal was to codify music by breaking it down into quantifiable data. Amazingly enough, every single song in the Genome's library (there are over 700,000 with roughly 10,000 new songs being added each month) is still analyzed by a Pandora employee, referred to as a "musicologist." These musicologists have a strong background in music theory, whether they were professional musicians, professors, or just passionate lovers of music. Musicologists take each song, and reduce and rate over 400 attributes of every song. [17] These attributes range from complicated, mathematical things, such as the rhythmic and key changes, to more heuristic and subjective things, like the "soulfulness" of the vocals.

After perfecting his Music Genome Project, Westergren lauched Pandora Internet Radio with a computer scientist and a mathematician. While Westergren spearheaded the ideas, the other two men built the technology and assembled and managed the software engineers. As of January 31, 2012, Pandora boasted over 125 million registered users. [11] The basic idea behind the company is relatively simple: choose a song or an artist that you like, and Pandora then creates and streams a playlist of similar songs based on your selection. It alls informs you why each song is appearing on your station. For example, I wanted a music station based off of the artist Ray LaMontagne. Pandora first picks a song by Ray LaMontagne, and then tells me this: "To start things off, we'll play a song that exemplifies the musical style of Ray LaMontagne which features modern R+B stylings, blues influences, mild rhythmic syncopation, extensive vamping and mixed acoustic and electric instrumentation." By first selecting a song by your specified artist, Pandora makes sure that it's (probably) something that you'll like. When the first song by Ray ends, Pandora played the song "Crazy Love" by Van Morrison, and notified me: "From here on out we'll be exploring other songs and artists that have musical qualities similar to Ray LaMontagne. This track, 'Crazy Love' by Van Morrison, has similar mellow rock instrumentation, blues influences, folk influences, a subtle use of vocal harmony and call and answer vocal harmony (antiphony)." Perhaps one of the most amazing parts about Pandora is that it works with all genres. My favorite example is when Raekwon comes on my friend's Ghostface Killah station, and Pandora explains its reasoning for the song's inclusion is: 'because it features east coast rap roots, gangsta rap attitude, R+B influences, funk influences and danceable beats.' [10] While musicologists analyze and attempt to quantify 400 attributes of songs, it definitely includes very subjective characteristics, such as "gangsta rap attitude" and "danceable beats," which are seemingly impossible to quantify.

Another characterstic of Pandora that makes it unique is that it was the first of its

kind in internet radio. Streaming internet radio is an ever-growing trend around the world, and Pandora is certainly in the lead as far as brand recognition goes, but also as profits go. Westergren and colleagues at times had to go bankrupt in order to keep the startup afloat, and the company nearly failed several times. This is largely contributed to the fact that Westergren and colleagues had no business model from which to base their company off. The combination of that and the licensing fees that all internet radio businesses must pay in order to stream music for free. [16] Pandora and other internet radio services actually pay more for licensing than broadcast or satellite radios pay. An upside to this is that by offering musical recommendations to its users (unlike broadcast or satellite radio), Pandora is able to track how much its users spend on purchasing music. In fact, in a 2009 interview, Westergren confirmed that at that time, about 45 percent of listeners are buying more music than they did before they used Pandora's service, and that Pandora sells over one million dollars worth of music monthly (on iTunes, Amazon, and other such sites). Westergren claims that Pandora is "one of the biggest music affiliate referral sites on the Internet."

Pandora is looking into expanding its outreach by moving towards automboile streaming. Currently, most cars come equipped with an auxiliary cable, in which users can plug in their iPhone or iPod and stream Pandora via 3G. Westergren plans on installing Pandora as a dashbaord application, which promotes safety by preventing users from having to fiddle around with their phones while driving. [14] This is in response to the number of users which are now using Pandora on a mobile phone. In 2009, data showed that over half of the daily 65,000 new registered users were doing so using a mobile phone. Since the spread of internet streaming capabilities on cell phones, mobile listening has shot from nothing to 35 percent of total listening time (back in 2009). Actually, half the people using Pandora on an iPhone by that year were listening to it in their car. Because of this, Westergren is very dedicated to the idea of a Pandora dashboard application to increase safety of all users. Although it has been five years without a launch of this yet, rest assured that Westergren and his team are working hard at this noble venture.

As a systems scientist that has always had an intense interest in music, I tried to create a scaled-down version of one of these music organization systems, focusing on only one musical factor: tempo. Webster's Dictionary defines tempo simply as "the speed at which a musical piece is played or sung." However, as a musician, it is relatively easy to notice that sometimes, two songs with the same number of beats per minute (abbreviated as BPM throughout this paper) can sound very different in regard to speed. This is because there is a difference between "written tempo" and "perceived tempo." [5] This phenomenon is known as hypermeter, which is the grouping of measures in which the measure itself serves as a beat. For the less musically inclined, hypermeter is "the perception of smaller metrical divisions combining to form larger metrical divisions." [4] Instead of processing each individual beat, your brain combines a series of beats together and processes it as just one beat, which simplifies how you hear the music.

Beats per minute are most commonly measured by a device called a metronome. This device produces a fixed, consistent ticking noise which represents an "aural pulse." Metronomes allow you to set a specific tempo, which usually ranges from 40 to 208 BPM. [8] The reason that metronomes do not generally go higher than this is because beyond around 200 BPM, the brain actually begins to perceive the beats differently. Instead of processing each individual beat, it now hears "beats" as subdivisions of larger beats - creating hypermeter.

Westergren himself admits that computers are not yet entirely accurate in recognizing

and assigning attributes to music. They are best at identifying where a downbeat falls and tempo, but in general, he believes that they "struggle to categorize music in the way a human ear can." [22] This is where his musicologists come in, and where my idea for evaluating perceived tempo of the human ear based on fuzzy lingusitic variables.

# 4 Background on the Approach

I intend to use a combination of neural networks and fuzzy control as an approach to this project.

Artificial Neural Networks are a methodology of information programming that is meant to emulate a biological nervous system in a machine. The idea is to make computers more animalistic in thinking, and promote learning in general rather than in rote. Many researchers have focused on this nonalgorithmic approach to imitate the brain because they understand how very powerful the human brain is. "Human brains process visual and auditory information much faster than modern computers."

An important part of understanding how to model neural networks is understanding the brain works. Firstly, the name neural network is because they are based on neurons from the brain, which are nerve cells. They are an important foundation for our nervous system. Neurons, unlike other cells in the human body, are designed to transmit information throughout the body. Each neuron generally has dendrites extending from the cell. These dendrites are treelike extensions of the cells that receive inputs from other neurons. These inputs travel from one cell to another using synpases, which are connections between the neurons. There is a firing process through these synapses from the nerve cells to transmit information from one cell to another, received by the dendrites and then the neurons can process the information. [21]

In artificial neural networks, modeling this behavior of our brains is done with two important concepts: the activation function, which imitates the firing from the synapses, and the linear basis function, which imitates the sum of all the activations received. In order to produce activation (and thus convert to firing) in the actual neurons in our brain, the length of the dendrite processes the summation of all the received activations. In most cases, this activation is modeled in an artificial neural network with a simple linear function, where the firing rate is proportional to the activation. Basically, to model a real neuron, we want to produce a given output based on a particular input. In order to do this, synaptic weights need to be set up appropriately.

The nervous system of humans is considered to be a connectionist system, as opposed to conventional digital computers, which have parallel distribute processes (PDPs) which are programmed only with algorithms. The idea behind neural networks are that there are many comparisons in both structure and function of PDPs and connectionist systems. Functionally, they both store and process information and make decisions. Structurally, they both transmit information via electrical signals and have "memories." They use these memories to store procedural knowledge and data. [2]

However, there are also fundamental differences in function and structure between PDPs and connectionist systems. Functionally, PDPs are very rapid and accurate at arithmetic calculation, but connectionist systems learn by example and experience instead. They can distinguish general patterns and learn and recall information based on association, which parallel distribute processes cannot do. Structurally, these two are also very different. PDPs store each piece of information in a single, distinct part of their

memories. They can only recall information by its "address" in the computer, while connectionist systems recall information by association, and store each piece of information across all or several parts of their memories. Although PDPs are very rapid and accurate at arithmetic calculation, they can only perform one operation at a time in the CPU. [2] Conversely, connectionist systems perform many operations simultaneously all over. While parallel distributive processes also do not have a mechanism for learning except by rote in the form a new program, connectionist systems learn by adjusting the strength of connections between various neurons in the brain. The key concept to take away from this is that the differences in functionality relate to the differences in structure. The main idea behind this is that by programming computers to learn more like brains than like machines, computers can act more like living nervous systems.

There are many different categories of artificial neural networks, and one categorization is to differentiate by how they learn. Supervised learning artificial neural networks learn general patterns from examples or historical data where each example has both an output and an input. This category of ANNs are used on both regression and classification applications. In this case, the resulting networks must have adjustable parameters that are updated by a supervised learning rule. [2] Because we are interested in shaping the input-output mappings of the networks according to a given training data set, supervised learning networks can also be called mapping networks. On the other hand, unsupervised learning artificial neural networks are given examples of historical data cases with only inputs and no outputs. Then they group these cases into meaningfully similar structures. This category is used only classification applications only. Without any feedback from the environment or pre-determined categories, the learning system detects and organizes repeated properties of the data. With this kind of learning, different parts of the networks are used to respond to detected different input patterns. [13] The network is generally trained to strengthen the firing to respond to regularly detected patterns in the data. The most common applications of unsupervised learning is data clustering, feature extraction, and similarity detection. The main idea of unsupervised learning is to determine whether there are natural groupings of data.

It's appropriate to to use supervised learning because this is a classification type of problem - I want to classify the songs into appropriate (similar) playlists. I wanted to determine the specific classes (where each class is a playlists of similar-percieved-speed songs) in advance, which are the five different fuzzy linguistic terms that people used to decsribe each song - very slow, slow, average, fast, and very fast. The two attributes I will be basing our groups on will be actual beats per minute and perceived beats per minute (determined by a fuzzy relation).

Fuzzy set theory is essentially a way to impersonate the vague aspects in human thinking. Using fuzzy logic provides us a mechanism for representing the human tendency to make use of vague concepts in our intuitive understanding of the functioning of complex systems. Fuzzy logic is best used in problems where we desire to emulate heuristic knowledge, or people's thinking. Like neural networks, fuzzy logic intends to imitate the brain in a way. "Human brains interpret imprecise and incomplete sensory information provided by perceptive organs." [2] Lotfi A. Zadegh, born in Azerbaijan but educated in Iran and the United States, founded fuzzy set theory as a field of study when he published his paper in 1965, titled Fuzzy Set Theory. [23] In 1973, he published a paper on fuzzy logic. Because of these publications, he is considered the father of fuzzy logic and fuzzy set theory. Today, Lotfi is revered by engineers worldwide, which is somewhat of a surprise to him. In an interview with Azerbaijan International in 1994, Lotfi himself

expressed some confusion and shock that so few social scientists have discovered how useful fuzzy set theory can be, as he originally expected people in the social sciences-economics, psychology, philosophy, linguistics, politics, sociology, religion and numerous other areas to pick up on it. He has found it interesting that fuzzy logic has been embraced in industry by incorporating it into "smart" consumer products. [9] Some of the most popular applications of fuzzy logic in consumer products are fuzzy rice cookers, which cook your rice perfectly, and improvement on focus in camera lenses. In this same interview, he also claimed that he saw the future of fuzzy logic interrelated with neural networks and genetic algorithms, or what we call as a collection, "soft computing." It seems as though Lotfi was correct, because twenty years later, here I am in a course called Soft Computing, learning about fuzzy logic, neural networks, and genetic algorithms, and how all three are interrelated.

The most common argument against fuzzy logic or fuzzy control is that it is subjective and arbitrary. These two terms in the English language are often commonly used interchangeably, but here I argue that subjective and arbitrary have two very different meanings. [18] According to the Merriam-Webster dictionary, subjective is defined as "relating to the way a person experiences things in his or her own mind; based on feelings or opinions rather than facts," while arbitrary is defined as "not planned or chosen for a particular reason; not based on reason or evidence." I argue that while fuzzy logic is subjective, it is not arbitrary. When definition something subjectively, two people may disagree in definition, but can understand why the other person defined it in the way that they did. However, defining something arbitrarily is very different, and much more closely associated with randomness. Fuzzy logic and fuzzy control are intended to be subjective.

There are various approaches to fuzzy logic control (FLC) systems, with the most popular being Mamdani, Tsukamoto, Sugeno and Larsen. [15] Fuzzy inference systems are intended to effectively model human expertise in a specific application through a selection of if-then rules. For this project, I will focus on the Mamdani-style Fuzzy Inference System. Ebrahim Mamdani, who taught as a professor at Imperial London University, was a scholar in the fields of artificial intelligence, electrical and electronic engineering, and human-computer interaction. [19] He did research in the 1970s on potential applications of fuzzy set theory. In his doctoral thesis, he did research in an attempt to determine whether computers can learn to perform a task by observing a human performing it. In 1974, he attempted to use fuzzy logic as a means to control a steam engine and boiler combination using the heuristic knowledge of experienced human operators and a set of linguistic control rules derived by the operators. [20] This is generally considered the first application of fuzzy set theory, and was actually ten years after fuzzy logic was invented by Lotfi Zadegh.

He summed up this inference system with four basic steps. The first is fuzzification of inputs; given crisp numerical inputs; this step is meant to fuzzify the inputs in a more linguistic way so that we can apply the inference rules to them. The second step is generally called fuzzy inference. In this step, we apply the actual linguistic inference rules, or control rules, to the fuzzy linguistic input values that we got from the first step. This step takes these fuzzy linguistic inputs and transforms them into fuzzy linguistic output values. The third step is called fuzzy output quantification, and is slightly more complicated than the first two steps. In this step, we take we fuzzy linguistic output values from our last step and convert them into fuzzy numerical outputs. These fuzzy numerical outputs are based on the output variable fuzzy relations. The most significant part of

this step is what is called the Max-min rule of Fuzzy Composition.....Finally, in our last step, we have the defuzzification of outputs. In this step, we take our fuzzy quantitative outputs and turn them into crisp quantitative outputs. One common defuzzification scheme to use is referred to as center-of-gravity, and that is what we will use here.

Fuzzy control, and specifically this Mamdani-style Fuzzy Inference System is very applicable in this project as a way to measure the perceived speed of different songs. However, in this case of analyzing perceived tempo, instead of using a full Mamdani-style Fuzzy Inference System with four steps, it makes sense for us do only the last two steps. As our perceived fuzzy speed is already being input in the form of a fuzzy linguistic variable, our main job here is to take these fuzzy linguistic inputs and output variable fuzzy relations, (step 3) and then take our fuzzy quantitative outputs and turn them into crisp quantitative outputs (step 4).

The fuzzy inference system takes a structured knowledge approach to represent if-then rules, but it does lack the ability to adapt to changing external environments. Thus, by using fuzzy inference systems incorporated into neural network learning, we get neuro-fuzzy modeling. My main plan with this project was to determine how humans would classify songs linguistically and then put them in a playlist (i.e. the songs that were generally decided as "very-slow" would all go in one playlist, and so on) and then train the neural network to learn this detection pattern. Then, given our validation data, testing the neural network to see if it can accurately determine which classification would be most appropriate given the actual BPM and the perceived BPM.

# 5 Details of Implementation

I began this project by choosing 21 different songs and recording their actual BPMs. In doing research on BPM, I came across Dave Tompkins, a postdoctoral research fellow in the computer science department of the University of British Columbia in Canada. [3] Tompkins has created a music database that is searchable in a variety of ways, including by BPM. He has used the MixMeister Pro program in order to calculate the BPM of all his music. All of the songs that I chose were from his database.

The Mamdani-style fuzzy inference system that I designed for this project was a one input, one output system with the following term sets:

PS: Perceived fuzzy speed
CS: Calculated fuzzy speed
$T_{PS} = \{VF, F, A, S, VS\}$ (for "very fast", "fast", "average", "slow", and "very slow")

The first thing I did was to create a fuzzy relation for the perceived speed of songs. I chose eight different songs to become the boundaries of each fuzzy linguistic term: that is, a song that's definitely very slow, two songs that are definitely slow (to be the beginning and end boundaries of slow), two songs that are definitely average speed (the beginning and end boundaries of average), two songs that are definitely fast (the beginning and end boundaries of average), and a song that is definitely very fast, in my own opinion. Then, I determined BPMs that I considered to be somewhat slow, average, fast, very fast, etc., and this is how I created the fuzzy relation. Table 1 contains the songs at the boundaries (starts and ends) of what I consider to be "definitely" in each fuzzy linguistic category. Now, I will explain subjectively how I decided what the boundaries should be.

| Title | Artist | BPM | Boundary |
|---|---|---|---|
| 4 40 | Sound For Movement | 40 | Very slow end |
| Weightless | Macaroni Union | 60 | Slow start |
| I Feel Fine | The Beatles | 90 | Slow end |
| Payphone | Maroon 5 | 110 | Average start |
| Hungry Like the Wolf | Duran Duran | 130 | Average end |
| Vertigo | U2 | 150 | Fast start |
| Cupid's Chokehold | Gym Class Heroes | 160 | Fast end |
| Stand By Me | 4 The Cause | 190 | Very fast start |

Table 1: Boundaries to form fuzzy relation

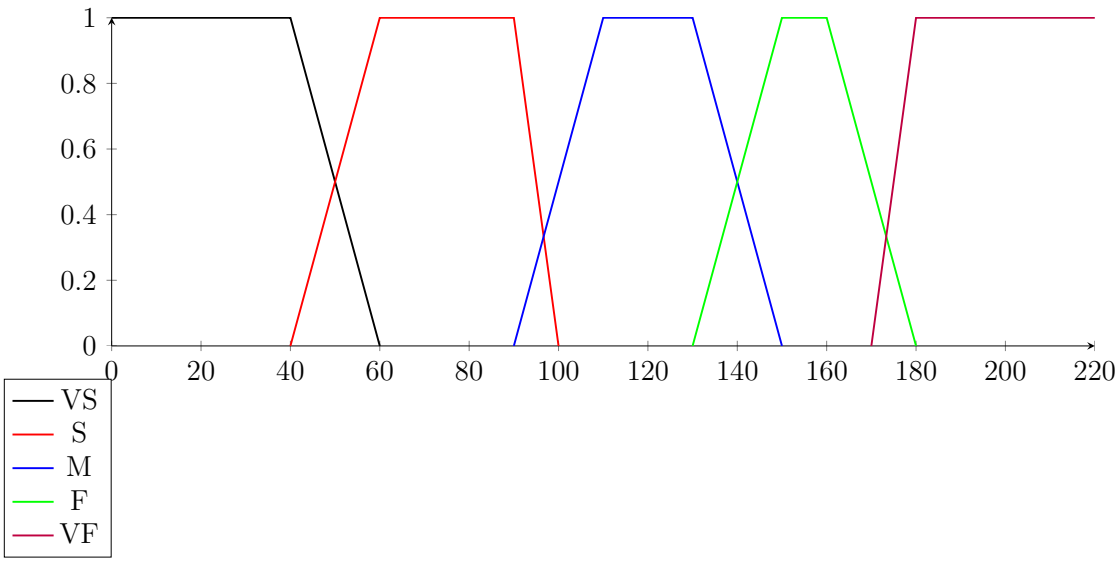



Figure 1: Fuzzy relation

I claim that songs with BPM from 0 (silence) to 40 are definitely considered "very slow." This is because most metronomes do not measure lower than 40 BPM. I claimed that songs with BPM from 60 to 90 were definitely considered "slow." "Average" speed songs are songs with BPMs from 110 to 130, and "fast" songs are from 140 to 150 BPM. I decided that any songs higher than 190 BPM would be considered "very fast", as most metronomes top out at 200 BPM, and this is when hypermeter begins, causing our brains to perceive the songs as half their actual speed. The resulting fuzzy relation, Figure 1, is based on those assumptions.

Then, I chose twenty-one songs from Dave Tompkins' database, with a wide range of BPM: the "slowest" song with a BPM of fifty and the "fastest" with a BPM of 174.8. I collected data by surveying peers and having them listen to each song. I then asked the to rate each song on a scale of "very slow" to "very fast." Just as I suspected, in most cases, the results were spread among several linguistic terms - there was not one song in which all of the 66 people that I surveyed agreed upon the same fuzzy linguistic term to describe the speed of the song.

At this point, I decided to integrate steps one and two of the Mamdani-style fuzzy inference system. The goal of step one is to take the crisp numerical inputs and fuzzify

them, and the goal of step two is to take these fuzzy linguistic inputs and turn them into fuzzy linguistic outputs based on the fuzzy linguistic rules. However, because I only have one input, and I collected data on people's heuristic knowledge of the speed of these songs, this input is already fuzzified. Although I skipped step one, I still have to do step two, which is coming up with a set of fuzzy inference rules to describe the output.

Therefore, instead of doing the traditional first step Mamdani-style fuzzy inference system, I determined the fuzzy lingusitic output for each song by taking a proportion of the given answers. See Table 2: Survey results.

| Song No. | VS | S | A | F | VF | Equation |
|---|---|---|---|---|---|---|
| 1 | 0 | 7 | 46 | 12 | 1 | $\frac{0}{VS} + \frac{0.11}{S} + \frac{0.70}{A} + \frac{0.18}{F} + \frac{0.02}{VF}$ |
| 2 | 0 | 1 | 7 | 48 | 10 | $\frac{0}{VS} + \frac{0.01}{S} + \frac{0.11}{A} + \frac{0.73}{F} + \frac{0.15}{VF}$ |
| 3 | 0 | 2 | 17 | 31 | 16 | $\frac{0}{VS} + \frac{0.03}{S} + \frac{0.26}{A} + \frac{0.47}{F} + \frac{0.24}{VF}$ |
| 4 | 0 | 8 | 27 | 25 | 6 | $\frac{0}{VS} + \frac{0.12}{S} + \frac{0.41}{A} + \frac{0.38}{F} + \frac{0.09}{VF}$ |
| 5 | 3 | 16 | 36 | 10 | 1 | $\frac{0.04}{VS} + \frac{0.24}{S} + \frac{0.55}{A} + \frac{0.15}{F} + \frac{0.02}{VF}$ |
| 6 | 3 | 14 | 29 | 19 | 1 | $\frac{0.04}{VS} + \frac{0.21}{S} + \frac{0.44}{A} + \frac{0.29}{F} + \frac{0.02}{VF}$ |
| 7 | 0 | 5 | 14 | 33 | 14 | $\frac{0}{VS} + \frac{0.08}{S} + \frac{0.21}{A} + \frac{0.5}{F} + \frac{0.21}{VF}$ |
| 8 | 0 | 2 | 20 | 30 | 14 | $\frac{0}{VS} + \frac{0.04}{S} + \frac{0.3}{A} + \frac{0.45}{F} + \frac{0.21}{VF}$ |
| 9 | 6 | 29 | 21 | 6 | 4 | $\frac{0.09}{VS} + \frac{0.44}{S} + \frac{0.32}{A} + \frac{0.09}{F} + \frac{0.06}{VF}$ |
| 10 | 28 | 28 | 10 | 0 | 0 | $\frac{0.42}{VS} + \frac{0.42}{S} + \frac{0.16}{A} + \frac{0}{F} + \frac{0}{VF}$ |
| 11 | 40 | 15 | 10 | 1 | 0 | $\frac{0.61}{VS} + \frac{0.23}{S} + \frac{0.15}{A} + \frac{0.01}{F} + \frac{0}{VF}$ |
| 12 | 18 | 31 | 13 | 4 | 0 | $\frac{0.27}{VS} + \frac{0.47}{S} + \frac{0.2}{A} + \frac{0.06}{F} + \frac{0}{VF}$ |
| 13 | 10 | 37 | 17 | 2 | 0 | $\frac{0.15}{VS} + \frac{0.56}{S} + \frac{0.26}{A} + \frac{0.03}{F} + \frac{0}{VF}$ |
| 14 | 3 | 17 | 42 | 3 | 1 | $\frac{0.05}{VS} + \frac{0.26}{S} + \frac{0.64}{A} + \frac{0.05}{F} + \frac{0}{VF}$ |
| 15 | 1 | 15 | 40 | 10 | 0 | $\frac{0}{VS} + \frac{0.26}{S} + \frac{0.61}{A} + \frac{0.13}{F} + \frac{0}{VF}$ |
| 16 | 0 | 7 | 43 | 16 | 0 | $\frac{0}{VS} + \frac{0.11}{S} + \frac{0.65}{A} + \frac{0.24}{F} + \frac{0}{VF}$ |
| 17 | 0 | 3 | 42 | 21 | 0 | $\frac{0}{VS} + \frac{0.04}{S} + \frac{0.64}{A} + \frac{0.32}{F} + \frac{0}{VF}$ |
| 18 | 0 | 0 | 4 | 46 | 16 | $\frac{0}{VS} + \frac{0}{S} + \frac{0.06}{A} + \frac{0.7}{F} + \frac{0.24}{VF}$ |
| 19 | 0 | 67 | 24 | 28 | 8 | $\frac{0}{VS} + \frac{0.09}{S} + \frac{0.36}{A} + \frac{0.42}{F} + \frac{0.13}{VF}$ |
| 20 | 0 | 14 | 32 | 15 | 5 | $\frac{0}{VS} + \frac{0.21}{S} + \frac{0.48}{A} + \frac{0.23}{F} + \frac{0.08}{VF}$ |
| 21 | 0 | 1 | 7 | 48 | 10 | $\frac{0}{VS} + \frac{0.04}{S} + \frac{0.17}{A} + \frac{0.56}{F} + \frac{0.23}{VF}$ |

Table 2: Survey results

Now for step two, I created a table to display the fuzzy inference rules. I put more weight into the perceived BPM than the actual BPM. Something worth mentioning that might look unusual at first is that when the actual BPM is "very slow" but the perceived BPM is "vey fast," I claimed that the song would be categorized as average. This is because of hypermeter, as mentioned in the background section of this paper. Most likely, the reason for the discrepancy between perceied BPM and actual BPM lies in the fact that our brains are perceiving there to be beats where they aren't actually, as if the song was actually twice its calculated BPM. The rule table is Table 3.

| | Actual BPM | | | | | |
|---|---|---|---|---|---|---|
| Perceived BPM | | VS | S | A | F | VF |
| | VS | SLOW | SLOW | SLOW | SLOW | AVG |
| | S | SLOW | SLOW | SLOW | AVG | AVG |
| | A | SLOW | AVG | AVG | AVG | FAST |
| | F | AVG | AVG | FAST | FAST | FAST |
| | VF | AVG | FAST | FAST | FAST | FAST |

Table 3: Rule Table

Now, I am able to begin step three, implemented like the Mamdani-style fuzzy inference system. Step three, fuzzy output quantification, is meant to take the fuzzy lingutic outputs from the previous step and convert them into fuzzy numerical outputs, based on the output variable fuzzy relations. This is where I apply the max-min rule of fuzzy composition. In this situation, the Max-Min Rule of Fuzzy Composition says:

$$\mu_{PS"}(y) = Max_{t \in T_{PS}} \left\{ min \left[ \mu_{PS'}(t), \mu_{R_{PS'}}(t, y) \right] \right\} \forall y \in Y_{PS}$$

Recall that $T_{PS} = \{VS, S, A, F, VF\}$ is the set of all linguistic terms associated with this variable, and $Y_{PS} = [0, 200]$ is the set of all numerical values associated with the variable, in other words, the possible BPMs. So for each possible numerical value $y \in [0, 200]$ for PS, I calculate $\mu_{PS"}(y)$ as shown in the equation. We are using $t$ to represent one of the possible linguistic terms. So at any given moment, $t$ can represent $VS, S, A, F, VF$. In the same way, $y$, as we already said represents the particular value from $y_{PS}$ that is being considering at the moment. Thus, at any particular moment, $y$ can be any real number from zero to two hundred. The notation $\mu_{PS'}(t)$ we know just means the membership degree by which term $t$ was represented in our results from step two. Continuing with our first song as the exmaple, we know $\mu_{PS'}(VS) = 0, \mu_{PS'}(S) = 0, \mu_{PS'}(A) = 0, \mu_{PS'}(F) = 0, \mu_{PS'}(VF) = 0$. By the notation $\mu_{R_{PS}}(t, y)$, what I mean is how well does that particular term $t$ relate to that particular numerical value $y$ according to the way we defined $R_{PS}$ when we first designed the system. Just as one example out of the infinite examples we could consider, note that we might say $\mu_{R_{PS}}(S, 62) = 0.8$, in other words, the linguistic term S (slow) relates quite well (to degree 0.8) to the numerical value 62, i.e. the song feels or sounds about 80 percent similar to songs with 62 BPM.

Now, what are the "Max" and "min" parts of the equation telling us to do? For a particular value of $y$, we are supposed to look at all (in this case all five) possible values of $t$, in each case taking the minimum of $\mu_{PS'}(t)$ (i.e. that term's membership grade in the results from step two) and $\mu_{R_{PS}}(t, y)$ (i.e. how well that term relates to that numerical value in our fuzzy relation). Then we take the maximum of those five minima. We repeat this calculation across all values of $y \in Y_{PS}$, and we'll need to do this for each of the 25 songs. I will illustrate a step-by-step example with the very first song, and we'll skip over repeating the same processing for the other 24 songs.

As you recall, we need to perform the min-max rule for all values of $y \in Y_{PS}$, but in this case $Y_{PS}$ is an infinite set, we can't explicitly enumerate them all. This might make it all the more convenient to think of this step graphically at first. Let us now examine the graphical interpretation of what all the maximizing and minimizing in the Max-min rule. We start by going back to look at the original graphical representation of this fuzzy relation.

First, consider what $min\left[\mu_{PS'}(t), \mu_{R_{PS'}}(t,y)\right]$ means if we consider the case of $t = VS$, and consider all possible values of $y$. As we can see from the graph $\mu_{R_{PS}}(VS,y)$ does take positive values where $0 \leq y < 40$, but that it always equals zero for $y \geq 40$. In any case, in the "results" we got from step two, $\mu_{PS'}(VS) = 0$, and naturally this is true no matter what $y$ is, so $min\left[\mu_{PS'}(VS), \mu_{R_{PS'}}(VS,y)\right] = 0$ for all $y \in [0,200]$. Getting to the term $t = S$ is where it gets more interesting. As you recall, for Song 1, $\mu_{PS'}(S) = .11$. From the plot for $R_{PS'}$, we see that

$$\mu_{PS'}(S,y) = \begin{cases} 0, & y \leq 40 \\ 0.05y - 2, & 40 < y \leq 60 \\ 1, & 60 < y \leq 90 \\ 10 - 0.1y, & 90 < y \leq 100 \\ 0, & y > 100 \end{cases}$$

So the question is, what is the minimum of $\mu_{PS'}(S)$ and $\mu_{R_{PS}}(S,y)$ for all the various values of $y$? A little analysis of the situation can convince us that up to $y = 42.2, \mu_{R_{PS}}(S,y)$ is the lower value, and then from $42.2 < y \leq 98.9$, $\mu_{PS'}(S) = .11$ is the lower value. Then, $\mu_{R_{PS}}(S,y)$ is once again the lower value for $y > 98.9$. Therefore, we can say:

$$min\left[\mu_{PS'}(S), \mu_{R_{PS}}(S,y)\right] = \begin{cases} 0, & y \leq 40 \\ 0.05y - 2, & 40 < y \leq 42.2 \\ .11, & 42.2 < y \leq 98.9 \\ 10 - 0.1y, & 98.9 < y \leq 100 \\ 0, & y > 100 \end{cases}$$

We continue this for $t = A$, which for Song 1, $\mu_{PS'}(A) = .7$. From the plot for $R_{PS'}$, we see that

$$\mu_{PS'}(A,y) = \begin{cases} 0, & y \leq 90 \\ 0.05y - 4.5, & 90 < y \leq 110 \\ 1, & 110 < y \leq 130 \\ 7.5 - 0.05y, & 130 < y \leq 150 \\ 0, & y > 150 \end{cases}$$

By some similar analysis, we determine that up to $y = 104, \mu_{R_{PS}}(A,y)$ is the lower value, and then from $104 < y \leq 136$, $\mu_{PS'}(A) = .7$ is the lower value. Similarly to $t = S$, $\mu_{R_{PS}}(A,y)$ is once again the lower value for $y > 136$. Therefore, our new piecewise function for this term is

$$min\left[\mu_{PS'}(A), \mu_{R_{PS}}(A,y)\right] = \begin{cases} 0, & y \leq 90 \\ 0.05y - 4.5, & 90 < y \leq 104 \\ .7, & 104 < y \leq 136 \\ 7.5 - 0.05y, & 136 < y \leq 150 \\ 0, & y > 150 \end{cases}$$

Carrying out the same process but without going into all the detail, for the final cases of $y = F$ and $y = VF$ gives us the following piecewise functions that represent the minimums of $\mu_{PS'}(t)$ and $\mu_{R_{PS}}(t, y)$.

$$min\left[\mu_{PS'}(F), \mu_{R_{PS}}(F, y)\right] = \begin{cases} 0, & y \leq 130 \\ 0.05y - 6.5, & 130 < y \leq 133.6 \\ .18, & 133.6 < y \leq 176.4 \\ 9 - 0.05y, & 176.4 < y \leq 180 \\ 0, & y > 180 \end{cases}$$

for $t = F$, and then for $t = VF$.

$$min\left[\mu_{PS'}(VF), \mu_{R_{PS}}(VF, y)\right] = \begin{cases} 0, & y \leq 170 \\ 0.1y - 17, & 170 < y \leq 170.2 \\ 0, & y > 170.2 \end{cases}$$

Now that we have completed the minimization part of step three, we need to consider for all values of $y \in [0, 200]$ the maximum of the minima across all five terms. Thinking about it a little makes it clear that for all values of $y$ up to 42.2, $min\left[\mu_{PS'}(S), \mu_{R_{PS}}(S, y)\right]$ will be greater than or equal to $min\left[\mu_{PS'}(t), \mu_{R_{PS}}(t, y)\right]$ for $t = VS, A, F, VF$, and then from $42.2 < y \leq 136$, $min\left[\mu_{PS'}(A), \mu_{R_{PS}}(A, y)\right]$ will have the greatest values over the other terms. Then, $min\left[\mu_{PS'}(F), \mu_{R_{PS}}(F, y)\right]$ has the greatest values from $136 < y \leq 180$, and clearly $min\left[\mu_{PS'}(VF), \mu_{R_{PS}}(VF, y)\right]$ is the max for $y > 180$, as the rest of the values in that threshold are 0. So we have,

$$\mu_{PS"}(y) = \begin{cases} 0, & y \leq 40 \\ 0.05y - 2 & 40 < y \leq 42.2 \\ .11 & 42.2 < y \leq 92.2 \\ 0.05y - 4.5 & 92.2 < y \leq 104 \\ 0.7 & 104 < y \leq 136 \\ 0.18 & 136 < y \leq 146.4 \\ 9 - 0.05y & 146.4 < y \leq 179.6 \\ 0.02 & 170.6 > y \end{cases}$$

Now, we'll need to complete this process for the 24 remaining example songs. You can find the details of this in Appendix II.

We are finally ready for step four, which defuzzifies the outputs by taking the fuzzy quantitative (numerical) outputs from the previous step and turns them into crisp quantiative outputs. These are the outputs that we will use to train our neural network, for that part of the project. Firstly, we awnt to consider the numerical value set $Y_PS$ to be the infinite set is actually is and consider this step analytically, we must implement simple integral calculus. As you recall from the last section, I hvae decided to use the COG (center-of-gravity) method. Specifically here, we are thinking of using the continuous form of COG. In this form, COG indicates that we calculate

$$PS = \frac{\int_{y \in Y_{PS}} y \cdot \mu_{PS"}(y)\, dy}{\int_{y \in Y_{PS}} \mu_{PS"}(y)\, dy}$$

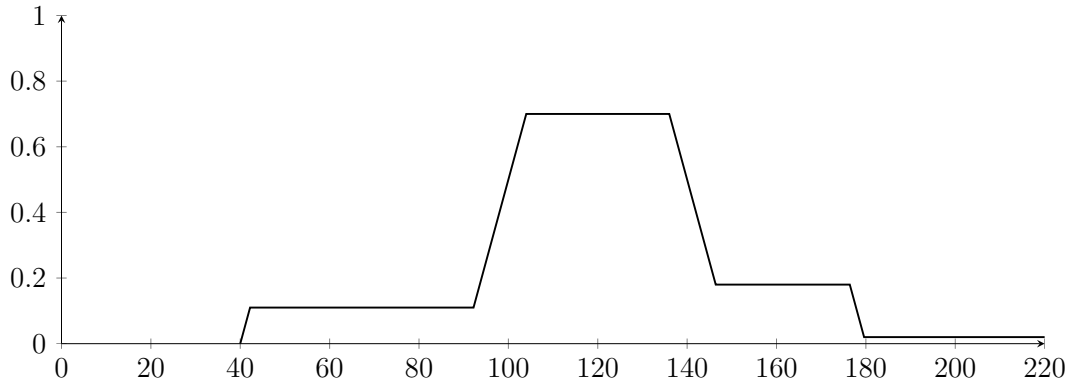



Figure 2: $\int_0^{220} \mu_{PS"}(y)\,dy$ graphically part 1

Calculating the denominator in this case tends to invoe basic geometric shapes from using the plot of the membership functions we get out of stepnumber three. Instead of going through the usual integral calculus approach, it might be less tedious to use basic geometry to how we calculate the denominator in the equation above. Starting with the denominator then, and continuing with our Song 1 example, let's calculate $\int_0^{220} \mu_{PS"}(y)\,dy$ where

$$\mu_{PS"}(y) = \begin{cases} 0 & y \leq 40 \\ 0.05y - 2 & 40 < y \leq 42.2 \\ .11 & 42.2 < y \leq 92.2 \\ 0.05y - 4.5 & 92.2 < y \leq 104 \\ 0.7 & 104 < y \leq 136 \\ 7.5 - 0.05y & 136 < y \leq 146.4 \\ 0.18 & 146.4 < y \leq 176.4 \\ 9 - 0.05y & 176.4 < y \leq 179.6 \\ 0.02 & 179.6 \geq y \end{cases}$$

Remembering the graphical interpretation of definite integral as "the area under the curve," we can do this in a simple way. See Figure 2.

The area under the curve (shaded-in area) in this particular case can be broken apart into eight basic geometric shapes.[1]

[1]To view the graphs, and calculate the area of these shapes efficiently, I used Geogebra 5, which is a multi-platform dynamic mathematics software that combines that brings together geometry, algebra, spreadsheets, graphing, statistics and calculus into a single, easy-to-use package. The open-source software has been winning Academic Software awards since 2003. You can find the graphs of each song listed in Appendix III.

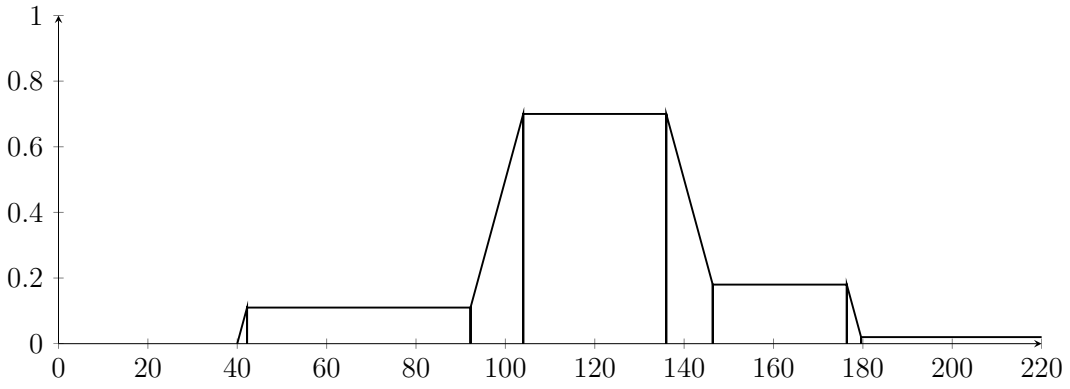



Figure 3: $\int_0^{220} \mu_{PS"}(y)\,dy$ graphically part 2

| | | | | | |
|---|---|---|---|---|---|
| 1) | a triangle whose base is | 2.2 | (=42.2-40) | and whose height is | .11 |
| 2) | a rectangle whose base is | 50 | (=92.2-42.2) | and whose height is | .11 |
| 3) | a trapezoid whose base is | 11.8 | (=104-92.2) | and whose height ranges from | .11 to .7 |
| 4) | a rectangle whose base is | 32 | (=136-104) | and whose height is | .7 |
| 5) | a trapezoid whose base is | 10.4 | (=146.4-136) | and whose height ranges from | .7 to .18 |
| 6) | a rectangle whose base is | 30 | (=176.4-136.4) | and whose height is | .18 |
| 7) | a trapezoid whose base is | 3.2 | (=179.6-176.4) | and whose height ranges from | .18 to .02 |
| 8) | a rectangle whose base is | 40.4 | (=220-179.6) | and whose height is | .01 |

It is then much simpler to calculate the area of each shape and sum these areas.

| | | | | | |
|---|---|---|---|---|---|
| 1) | (.5)bh | = | (2.2)(.11)(.5) | = | 0.242 |
| 2) | bh | = | (50)(.11) | = | 5.5 |
| 3) | $h\frac{b1+b2}{2}$ | = | $(11.8)\frac{.11+.7}{2}$ | = | 4.779 |
| 4) | bh | = | (32)(.7) | = | 22.4 |
| 5) | $h\frac{b1+b2}{2}$ | = | $(10.4)\frac{.7+.18}{2}$ | = | 4.576 |
| 6) | bh | = | (30)(.18) | = | 5.4 |
| 7) | $h\frac{b1+b2}{2}$ | = | $(3.2)\frac{.18+.02}{2}$ | = | 0.32 |
| 8) | bh | = | (40.4)(.02) | = | 0.808 |

Thus, we have $\int_0^{220} \mu_{PS"}(y)\,dy = 44.025$.

The numerator is a little bit tricker, but not much. Anytime we apply calculus to polynomials (or even, as in this case, piecewise polynomials) life is pretty easy. We need to find $\int_{y \in Y_{PS}} y \cdot \mu_{PS"}(y)\,dy$ where

$$y \cdot \mu_{PS"}(y) = \begin{cases} 0 & y \leq 40 \\ 0.05y^2 - 2y & 40 < y \leq 42.2 \\ .11y & 42.2 < y \leq 92.2 \\ 0.05y^2 - 4.5 & 92.2 < y \leq 104 \\ 0.7y & 104 < y \leq 136 \\ 7.5y - 0.05y^2 & 136 < y \leq 146.4 \\ 0.18y & 146.4 < y \leq 176.4 \\ 9y - 0.05y^2 & 176.4 < y \leq 179.6 \\ 0.02y & 179.6 \geq y \end{cases}$$

Skipping over the tedius calculations of the piecewise functions integrated between their min and max values, we find that $\int_{y \in Y_{PS}} y \cdot \mu_{PS"}(y)\, dy = 5624.77997$.

Now, finally, we can calculate,

$$PS = \frac{\int_{y \in Y_{PS}} y \cdot \mu_{PS"}(y)\, dy}{\int_{y \in Y_{PS}} \mu_{PS"}(y)\, dy} = \frac{5624.77997}{44.025} = 127.7633.$$

So, in terms of perceived speed, this song predicts that this song "feels" like 127.7633 beats per minute. Since the song feels most like 127.7633 beats per minute, according to our original fuzzy relation, we want to train the neural network to classify this song as "average." The way we would use this to train the neural network part would be to ensure that the output would be equal to 0.9 for class 3 (average), and less than 0.9 for the her classes (very slow, slow, fast, very fast). Results of this process for the remaining 24 songs are located in Appendix IV.

Now, we are ready to tackle the neural network half of this project. As discussed in the previous section, I used a simple kind of supervised learning artificial neural network with two input nodes, one being the actual BPM, and the other being the perceived BPM, which we just calculated for each song using the Mamdani-style Fuzzy Inference System. I decided that there would be five output nodes, which represent the five different "speeds" (very slow, slow, average, fast, very fast) In this case, like most unsupervised learning approaches, we'll be normalizing in the inputs by scaling them to a range of values from 0 to 1. Before we do that, however, we will preprocess by taking the logarithms (common logs) of the inputs first. Thus, scaled to the

$$x_r[1] = log\,(\text{actual BPM of song number r})$$
$$x_r[2] = log\,(\text{perceived BPM of song number r})$$

The range of both inputs will be approximately 0.0 to 2.3 scaled to the unit interval. The results in this simple algorithm depend somewhat on the initial weights. I decided to allow the machine to only produce 5 clusters (i.e. playlists) of similar-speed songs (very slow, slow, average, fast, very fast). Of the 21 songs, I used 2/3 of this data to train the neural network, which is p=14 songs. The other 7 songs are used for validation data, to ensure that the machine is properly classifying the songs.

Continuing with our example of Song 1, we know

$$x_1[1] = log\,(150.7) = 2.178113$$

$$x_1[2] = log(127.7633) = 2.106406$$

As previously mentioned, since this song resulted in an classification of an "average" speed song from our fuzzy inference rules, we want our $y_r$ results, and thus, our input table, to look like this:

| $y_r(2)$ | $y_r(2)$ | $y_r(3)$ |
|---|---|---|
| .1 | .9 | .1 |

Table 4: $y_r$ for Song 1

In this project, I have decided to use batch training with a typical learning factor of $\rho = 0.05$. Starting from ignorance for our weights, the values calculated by using a pseudo-random number generator and $\frac{4 \cdot rand - 2}{3}$ in order to keep the weights between -1 and 1.

| $w_1$ | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.4196 | 0.5411 | -0.4974 |
| 2 | 0.5512 | 0.1765 | -0.5366 |
| 3 | -0.2953 | 0.0625 | 0.6100 |

Table 5: $w_1$ randomized

Now we can begin training our neural network. The fourteen data points we previously said we would use (in their preprocessed forms) are located in Table 8.

| Song # | AS | PS | xr(1) | xr(2) |
|---|---|---|---|---|
| 1 | 150.7 | 127.7 | 2.178113252 | 2.106190897 |
| 2 | 126.1 | 156.6 | 2.100715087 | 2.194791758 |
| 3 | 150.7 | 149.5 | 2.178113252 | 2.174641193 |
| 4 | 104.6 | 131.2 | 2.019531685 | 2.117933835 |
| 5 | 136.7 | 108.4 | 2.135768515 | 2.035029282 |
| 6 | 125 | 119.3 | 2.096910013 | 2.076640444 |
| 7 | 63.5 | 147.8 | 1.802773725 | 2.169674434 |
| 8 | 128 | 147.8 | 2.10720997 | 2.169674434 |
| 9 | 142.5 | 100.8 | 2.153814864 | 2.003460532 |
| 10 | 65 | 60.9 | 1.812913357 | 1.784617293 |
| 11 | 74 | 50.4 | 1.86923172 | 1.702430536 |
| 12 | 88.9 | 88.9 | 1.948901761 | 1.948901761 |
| 13 | 82.2 | 73.4 | 1.914871818 | 1.86569606 |
| 14 | 90.4 | 96.8 | 1.95616843 | 1.985875357 |

Table 6: Training Data

Let's go through the first epoch as an example of how the algorithm works. For preprocessing $a_0$, we took the common logarithm of the actual speed and perceived speed, which are what you see in Table 8 as $x_r$. Then, we normalized our values to be between 0 and 1.

Initializing our c1 matrix of all zeros, we'll (continue to) consider our first training example. We know

| i | 0 | 1 | 2 |
|---|---|---|---|
| $a_0[i]$ | 1.000 | 0.929863923 | 0.899150659 |

Table 7: $a_0$ for Song 1

Now, we want to forward propagate, beginning with $j = 1$ and calculating $s$ and $a_1[1]$.

$$s = (0.4196)(1.000) + (0.5411)(0.929863923) + (-0.4974)(0.899150659) = 0.4755$$

$$a_1[1] = \frac{1}{1 + e^(-0.4755)} = 0.6167$$

Now, the same thing, for $j = 2$:

$$s = (0.5512)(1.000) + (0.1765)(0.929863923) + (-0.5366)(0.899150659) = 0.2328$$

$$a_1[2] = \frac{1}{1 + e^(-0.2328)} = 0.5579$$

Finally, for $j = 3$:

$$s = (-0.2953)(1.000) + (0.0625)(0.929863923) + (0.6100)(0.899150659) = 0.3113$$

$$a_1[3] = \frac{1}{1 + e^(-0.3113)} = 0.5772$$

So, now we have:

| $a_1(j)$ | | |
|---|---|---|
| 1 | 2 | 3 |
| 0.6167 | 0.5579 | 0.5772 |

Table 8: $a_1$ for Song 1

And now we can begin the back propagate portion, calculating $\epsilon, d$, and our new values for the $c1$ matrix.

$$j = 1$$

$$\epsilon = 0.6167 - 0.1 = 0.5167$$
$$d = (0.5167)(0.6167)(1 - 0.6167) = 0.1221$$
$$c_1[1, 0] = 0.0000 + (0.1221)(1.000) = 0.1221$$
$$c_1[1, 1] = 0.0000 + (0.1221)(0.929863923) = 0.1135$$
$$c_1[1, 2] = 0.0000 + (0.1221)(0.899150659) = 0.1098$$

$$j = 2$$

$$\epsilon = 0.5579 - 0.9 = -0.3421$$
$$d = (-0.3421)(0.5579)(1 - 0.5579) = -0.0844$$
$$c_1[2, 0] = 0.0000 + (-0.0844)(1.000) = -0.0844$$
$$c_1[2, 1] = 0.0000 + (-0.0844)(0.929863923) = -0.0785$$
$$c_1[2, 2] = 0.0000 + (-0.0844)(0.899150659) = -0.0759$$

$$j = 3$$

$$\epsilon = 0.5772 - 0.1 = 0.4772$$
$$d = (0.4772)(0.5772)(1 - 0.5772) = 0.1165$$
$$c_1[3, 0] = 0.0000 + (0.1165)(1.000) = 0.1165$$
$$c_1[3, 1] = 0.0000 + (0.1165)(0.929863923) = 0.1083$$
$$c_1[3, 2] = 0.0000 + (0.1165)(0.899150659) = 0.1048$$

So, now we have our new $c1$ matrix:

| c1 | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.1221 | 0.1135 | 0.1098 |
| 2 | -0.0844 | -0.0785 | -0.0759 |
| 3 | 0.1165 | 0.1083 | 0.1048 |

Table 9: $c_1$ matrix for Song 1

However, we don't want to update the weights yet, since this is batch training. I went through the other 13 training examples, and then at the end of the first epoch, we can finally adjust the weights for the first time.

$$\text{new } w_1[1, 0] = 0.4196 - (0.05)(0.1221) = 0.4135$$
$$\text{new } w_1[1, 1] = 0.5411 - (0.05)(0.1135) = 0.5355$$
$$\text{new } w_1[1, 2] = -0.4974 - (0.05)(0.1098) = -0.5029$$
$$\text{new } w_1[2, 0] = 0.5512 - (0.05)(-0.0844) = 0.5554$$
$$\text{new } w_1[2, 1] = 0.1765 - (0.05)(-0.0785) = 0.1804$$
$$\text{new } w_1[2, 2] = -0.5466 - (0.05)(-0.0759) = -0.5428$$
$$\text{new } w_1[3, 0] = -0.2953 - (0.05)(0.1165) = -0.3011$$
$$\text{new } w_1[3, 1] = 0.0625 - (0.05)(0.1083) = 0.0571$$
$$\text{new } w_1[3, 2] = 0.6100 - (0.05)(0.1048) = 0.6048$$

and thus, our new weights at the end of the first epoch are:

| w1 | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0.4135 | 0.5355 | -0.5029 |
| 1 | 0.5554 | 0.1804 | -0.5428 |
| 2 | -0.3011 | 0.0571 | 0.6048 |

Table 10: $w_1$ updated for Song 1

Lastly, I calculated the overall error function for each epoch. Our error function gives us a useful statistic on how the BP algorithm is doing in any given epoch. We can consider this a reflection of good (actually bad) the the previous weights were. Over time, the overall error function should go down epoch after epoch.

Now, it's time to use our validation data to see how well the model will actually work. We'll use Song 15 as an example of how to use our data to see how the well the model predicts the correct playlist. The actual BPM of Song 15 is 126.1, and the perceived BPM after using the fuzzy inference system is 108.8. Therefore, we ideally want the neural network to place this in the "average" playlist, meaning that $y_r[2] > y_r[1], y_r[3]$.

# 6 Description of Results

I calculated the overall error function for each epoch during training. Our error function gives us a useful statistic on how the BP algorithm is doing in any given epoch. We can consider this a reflection of good (actually bad) the the previous weights were. Over time, the overall error function should go down epoch after epoch. As previously mentioned, I trained for 100 epochs with a learning factor of $\rho = 0.05$. The below graph shows the error function over time over the 100 epochs. As you can see, it clearly decreases, showing that the network is training well and learning.

Now, it's time to use our validation data to see how well the model will actually work. We'll use Song 15 as an example of how to use our data to see how the well the model predicts the correct playlist. The actual BPM of Song 15 is 126.1, and the perceived BPM after using the fuzzy inference system is 108.8. Therefore, we ideally want the neural network to place this in the "average" playlist, meaning that $y_r[2] > y_r[1], y_r[3]$.

# 7 Conclusions

For the example of Song 15, the neural network determined that this song should be in the Average playlist, which is where the Rule Table classified it as well. However, for all of the validation data, the neural network only correctly calculated three out of the seven songs, which is 42.86% correct. The results of each specific song can be found in Appendix V. While the machine is incorrect about half of the time, it's actually not too bad, and I would predict that this percentage of correctly classified songs would increase with more training data. The most encouraging part is that it never classified a slow song (according to the rule table) as fast, nor a fast song (according to the rule table) as slow.

It seems that applying the idea of a fuzzy inference system and a neural network to determine playlists based on perceived speed was somewhat successful, as the machine correctly determined nearly half of the given songs. This is not a bad percentage, especially given that we only used 14 songs for our training data, and we only had seven

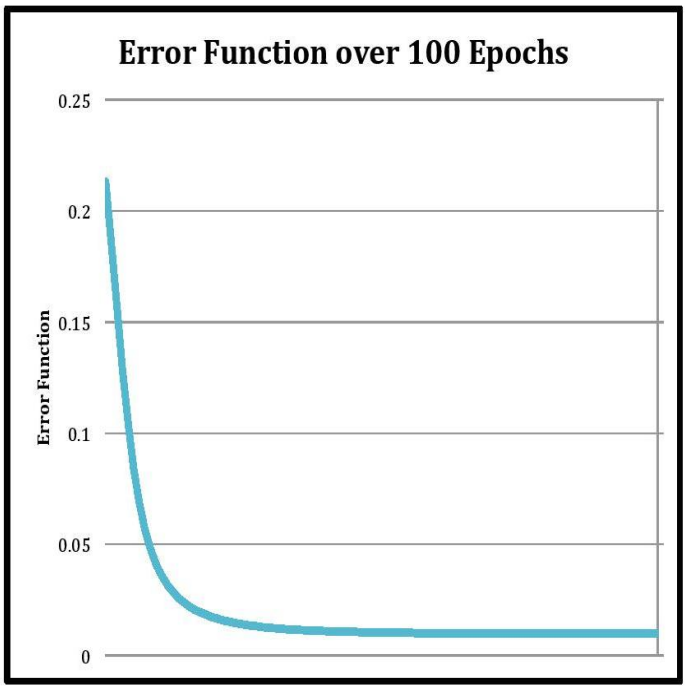



Figure 4: Error Function

songs to use for validation data.

One thing that I intended to implement with this project was calculating a given song's perceived speed by inputting the results of a survey, and then doing the Mamdani-style fuzzy inference system to determine the perceived fuzzy speed. I calculated the percved fuzzy speed using Excel and Geogebra, but if I worked on this project further, I would definitely like to see if there was a way that I could implement what I did using these programs by writing a MATLAB program that would do the same thing for me, without having to do it out by hand.

In the future, to extend or expand on this project, I would perhaps delve a little more deeply into what the most important connections are between songs in a playlist, other than speed. I think including genre as another input, in addition to percved speed, would be a good place to start. The most difficult part of this would determining meaningful connections between genre and speed, and which is more important in relating songs to each other. By determining the similarity of songs based on genre as well as perceived speed, music classification becomes a little more personalized. Although two songs could have the same perceived speed, most people would probably not incldue both rock and classical music together, or country and jazz music together. This, of course, would lead to having three input nodes in the neural network.

Decade is perhaps another potential characteristic of songs that could be included as another input node into determining what playlist a song should be included in. For example, in general, songs from the 1980's are much more "poppy" than many songs produced today. It would probably make sense for songs from the 80's to be grouped together in a playlist.

Another possible extention of this project (still somewhat related to different song characteristics) would be changing this from a supervised neural network to an unsuper-

vised neural network, and letting the machine determine meaningful playlists. I decided not to use unsupervised learning for this project, since the only two characteristics that were observed were speed-related, and decided that three playlists divided by speed made the most sense. However, if we added additional characteristics, it might make more sense to use unsupervised learning, and let the machine choose exactly how it wanted to group the songs by various characteristics (speed, genre, year, and so on).

# 8 Bibliography

# References


[1]

[2] Class notes.

[3] Dave tompkins.

[4] Hypermeasure.

[5] Why does some music sound faster than others even though its bpm is the same? 2013.

[6] About pandora media. 2014.

[7] Last.fm. 2014.

[8] Metronome. 2014.

[9] B. Blair. Interview with lotfi zadeh creator of fuzzy logic. *Azerbaijan International*, 2:46–47, 50, 1994.

[10] B. Christopher. The story of pandora radio. *Why didn't I think of that*, 2010.

[11] S. Clifford. Pandora's long strange trip. 2007.

[12] A. Diallo. Quest for the perfect playlist: itunes radio, pandora and spotify. *Forbes Magazine*, 2013.

[13] C. Sun J. R. Jang and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.

[14] D. Ballantyne W. Gotshall-Maxon J. Routley, L. Aranda and M. Miller. Pandora media.

[15] A. Kaur. Comparison of mamdani-type and sugeno-type fuzzy inference systems for air conditioning system. *International Journal of Soft Computing and Engineering (IJSCE)*, 2:323–325, 2012.

[16] M. Lasar. The perils of being pandora. 2011.

[17] J. Layton. How pandora radio works. 2006.

[18] H. Lewis. *The Foundations of Fuzzy Control*. IFSR International Sies on Systems Science and Engineering, 1997.

[19] E. Mamdani. Ebrahim mamdani. 2006.

[20] Ebrahim Mamdani. Ebrahim mamdani.

[21] M. Raichle and A. Snyder. A default mode of brain function: A brief history of an evolving idea. Neuroimage

[22] A. Salkever. Pandora's founder: We can make the internet radio business work. 2009.

[23] L. Zadegh. Dr. lotfi a. zadeh. *Scholarpedia*.

# 9 Appendices

## 9.1 Program Listings

```
function trainNeuralNet()
times = 100;
rho = 0.05; %learning factor
p = 1433; %number of training data points we're using
n = 2; %num of input nodes
m = 3; %num of output nodes
c1 = zeros(3, 3);
w1 = zeros(3, 3);

    fprintf ('Wait one moment please...training.');
    disp(' ');

    filename = 'data1.xlsx';
    xr = xlsread(filename);
    % reads data from the first worksheet in the Microsoft® Excel® spreadsheet file
    % named filename and returns the numeric data in array data.

    filename = 'data2.xlsx';
    yr = xlsread(filename);
    % reads data from the first worksheet in the Microsoft® Excel® spreadsheet file
    % named filename and returns the numeric data in array data.

    for t = 1:times %epochs

        if (t == 1)
            %initialize weights randomly if first epoch
            for j = 1:3
              for i = 1:3
                  w1(j,i) = (4*rand()-2)/3; %random numbers from -2/3 to 2/3
              end
          end
      else
          %w1 stays the same from the previous epoch instead of generated randomly
      end

      %initialize / zero out the partial accumulators
      for j = 1:3
          for i = 1:3
              c1(j, i) = 0.0;
          end
      end
```

```
        %begin training
        for r = 1:p

            a0 = zeros(1, 3);
            %read in example #r
            for i = 2:3
                a0(1) = 1.0;
                a0(i) = log10(xr(r,i-1));
            end

            %normalize to between 0 and 1
            for i=2:3
                a0(i) = (a0(i))/(log10(220));
            end

            s = 0;
            sSUM = 0;
            a1 = zeros(1, 3);
            %forward propagate
            for j = 1:3
                for i = 1:3
                    s = w1(j,i)*a0(i);
                    sSUM = s + sSUM; %summation
                    a1(j) = 1/(1+exp(-s));
                end
            end
    eps = zeros(1, 3);
    %backward propagate
    for j = 1:3
        eps = a1(j)-yr(r,j);
        e1(j) = eps;
        d = eps*a1(j)*(1-a1(j));
        for i = 1:3
            c1(j, i) = c1(j, i) + d*a0(i);
        end %end i
    end %end j
 end %end r

 %end of the epoch, now we can adjust weights
 for j = 1:3
     for i = 1:3
         w1(j,i) = w1(j,i) - rho*c1(j,i);
     end
 end

%save weights to a file
filename = 'weights2.xlsx';
xlswrite(filename,w1);
```

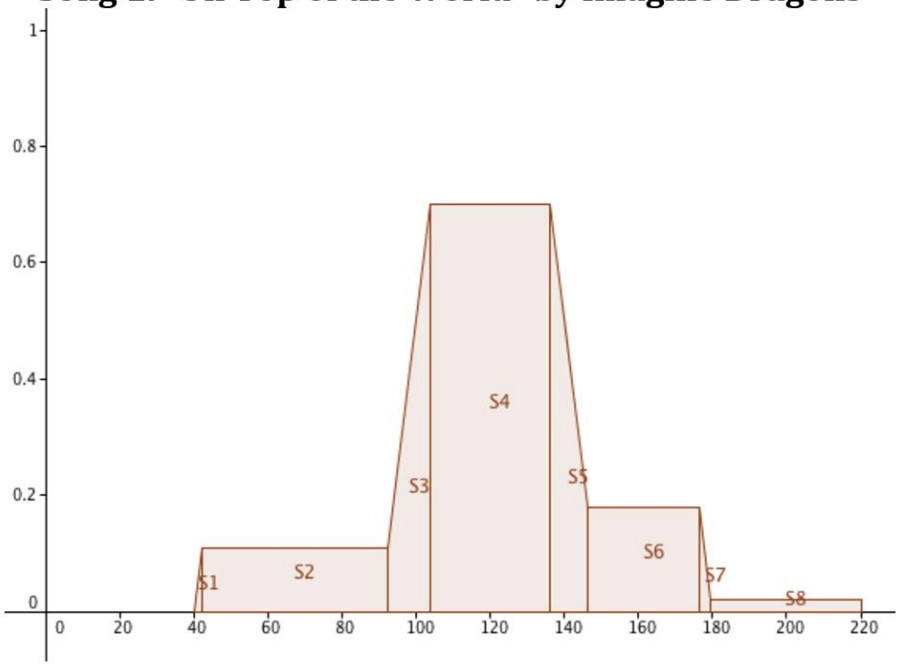
```
        %overall error function
        error = 0;
        for r = 1:p
            for j = 1:3
                error = (.5)*e1(j)^2;
            end
        end
    end %end t

    fprintf('Training complete!\n')

end
```

```
function validationData(actualBPM, perceivedBPM)
n = 2; %num of input nodes
m = 3; %num of output nodes

    preprocessedData = [actualBPM, perceivedBPM];

    %load weights from trained neural network
    filename = 'weights2.xlsx';
    w1 = xlsread(filename);

    %preprocessing
    %taking the (common) log and scaling to be between .1 and .9 for
    %activation
    a0 = zeros(1, 3);
    a0(1) = 1;
    a0(2) = log10(actualBPM);
    a0(3) = log10(perceivedBPM);

    for i=2:3
        a0(i)=(a0(i))/(log10(220));
    end

    %linear basis function
    %should be three nodes in s, because three output nodes
    s = zeros(1, 3);
    s(1) = w1(1,1)*a0(1) + w1(1,2)*a0(2) + w1(1,3)*a0(3);
    s(2) = w1(2,1)*a0(1) + w1(2,2)*a0(2) + w1(2,3)*a0(3);
s(3) = w1(3,1)*a0(1) + w1(3,2)*a0(2) + w1(3,3)*a0(3);

%calculate activation
a1 = zeros(1, 3);
for k = 1:3
    a1(k) = 1/(1+exp(-s(k)));
end

%postprocessing
index = find(a1 == max(a1(:)));
if (index == 1)
    speed = 'slow';
elseif (index == 2)
    speed = 'average';
else
    speed = 'fast';
end
fprintf('Your song belongs in the %s playlist.\n', speed);
end
```

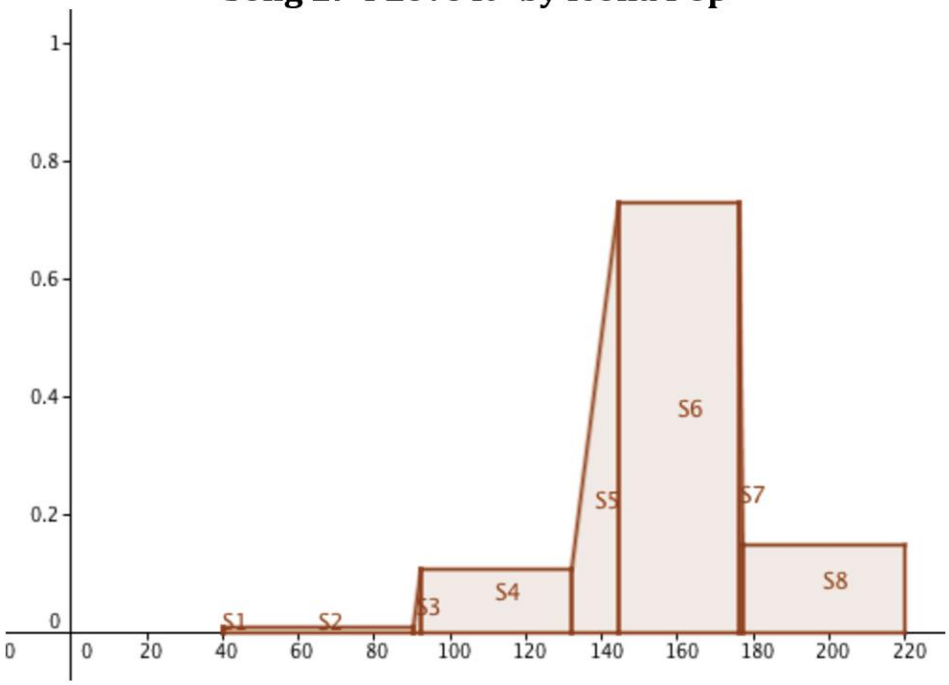
```
function musicClassifier()

    fprintf('Welcome to my neuro & fuzzy approach to a music classification system.\n');
    prompt2 = 'Please enter the actual BPM of the song.\n';
    actualBPM = input(prompt2);
    prompt3 = 'Now, please enter the perceived BPM of the song.\n';
    perceivedBPM = input(prompt3);

    trainNeuralNet();
    validationData(actualBPM, perceivedBPM)

end
```

## 9.2 Geogebra Graphs

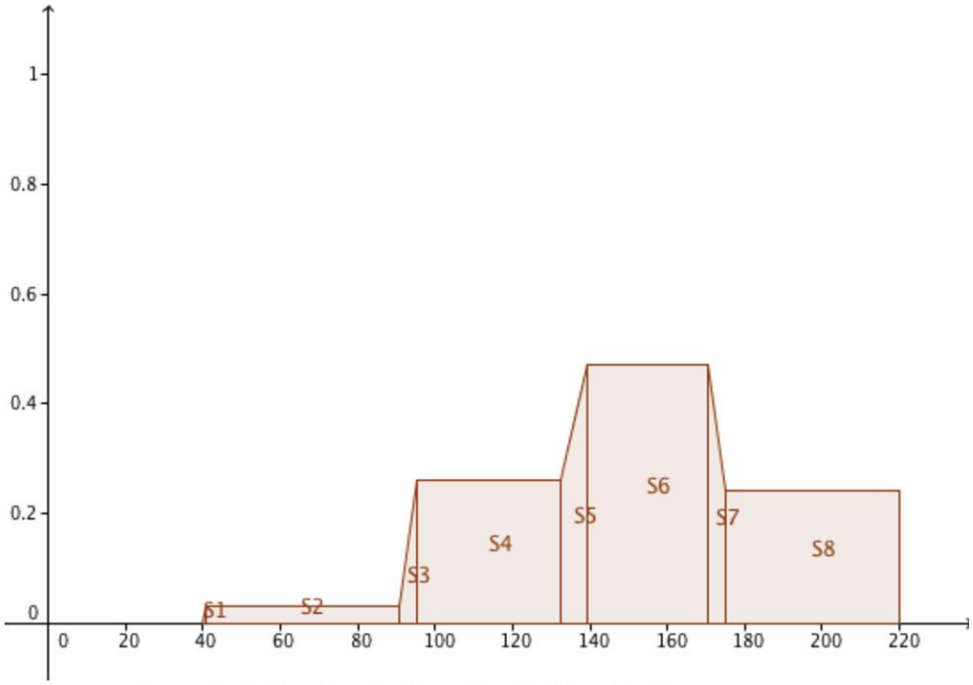
**Song 1: "On Top of the World" by Imagine Dragons**



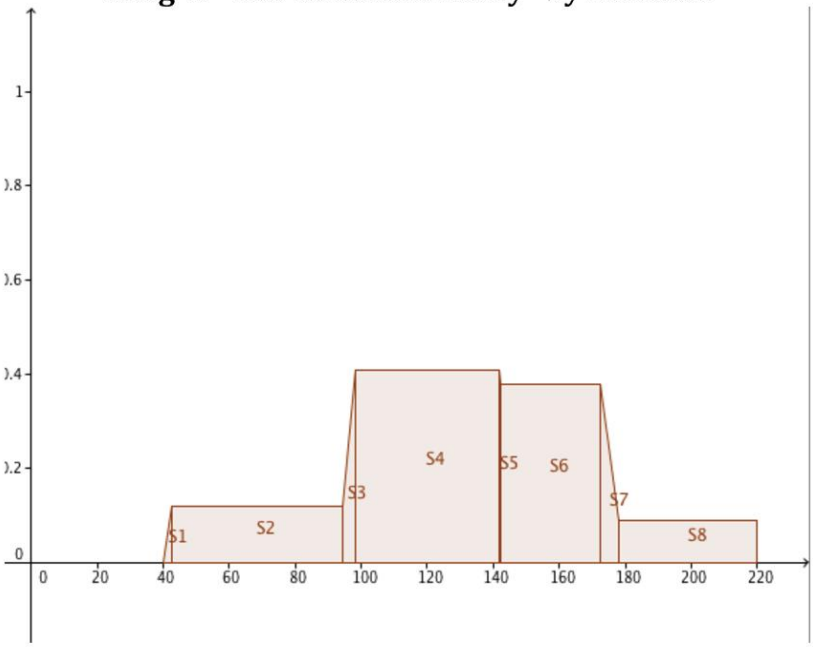

**Song 2: "I Love It" by Icona Pop**



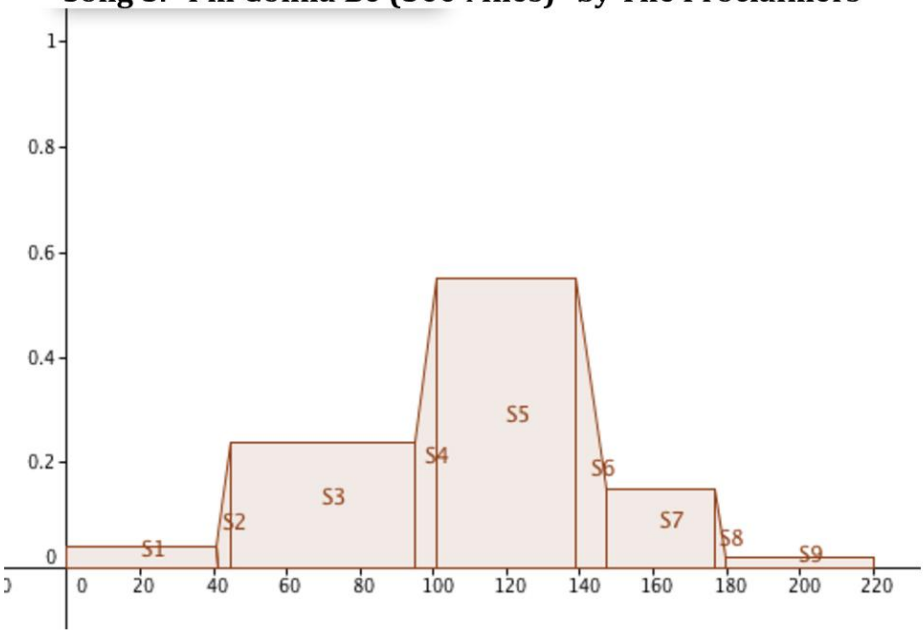
**Song 3: "Entertainment" by Phoenix**



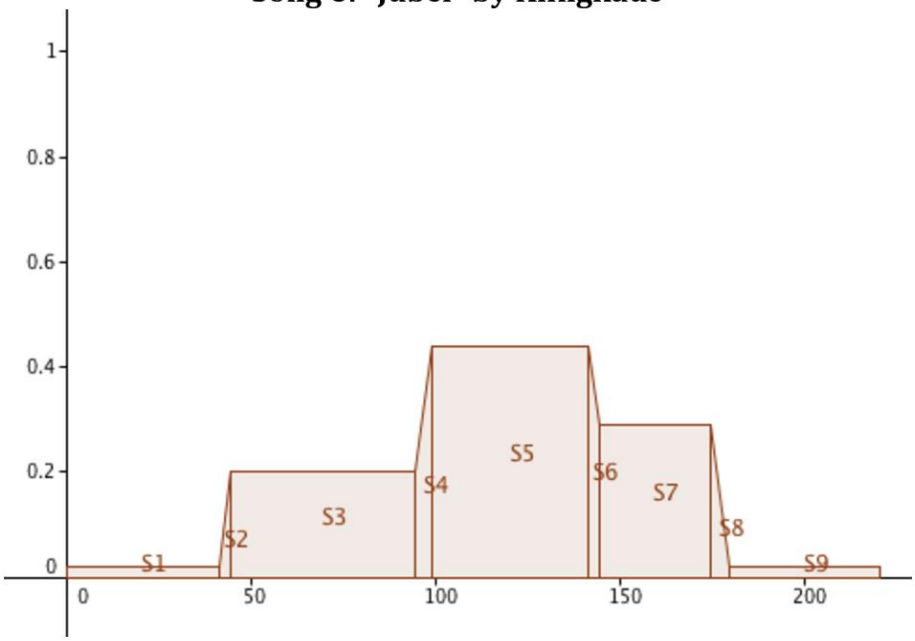

**Song 4: "The Real Slim Shady" by Eminem**



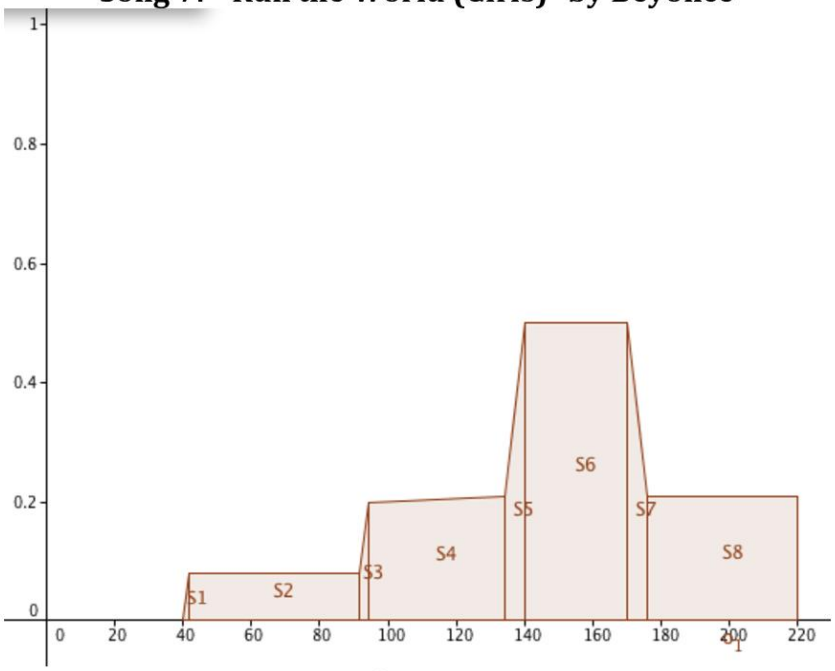
**Song 5: "I'm Gonna Be (500 Miles)" by The Proclaimers**



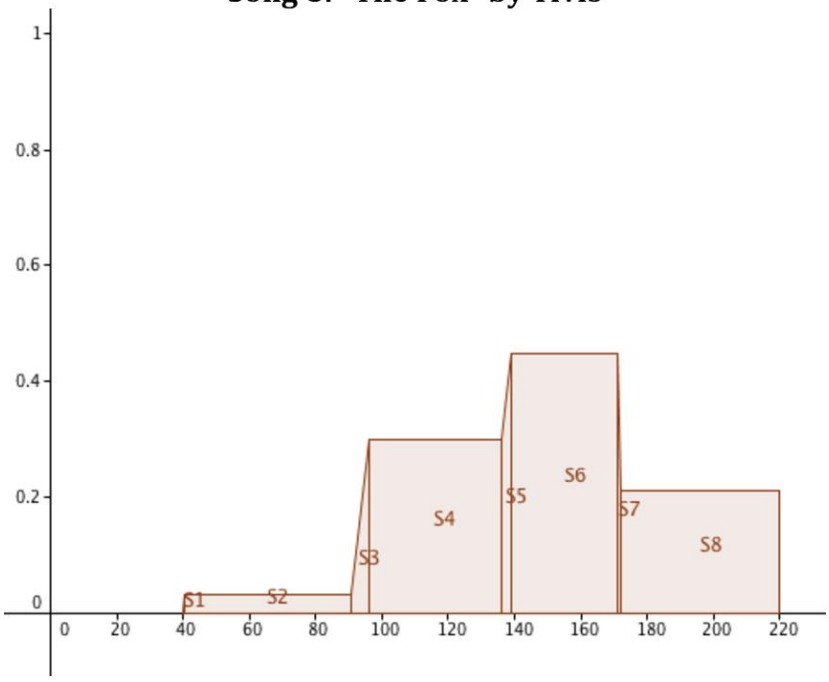

**Song 6: "Jubel" by Klingkade**



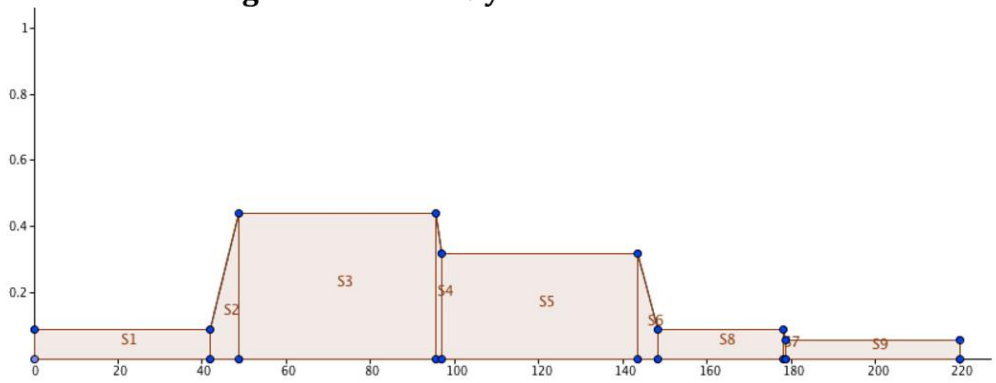
**Song 7: "Run the World (Girls)" by Beyonce**



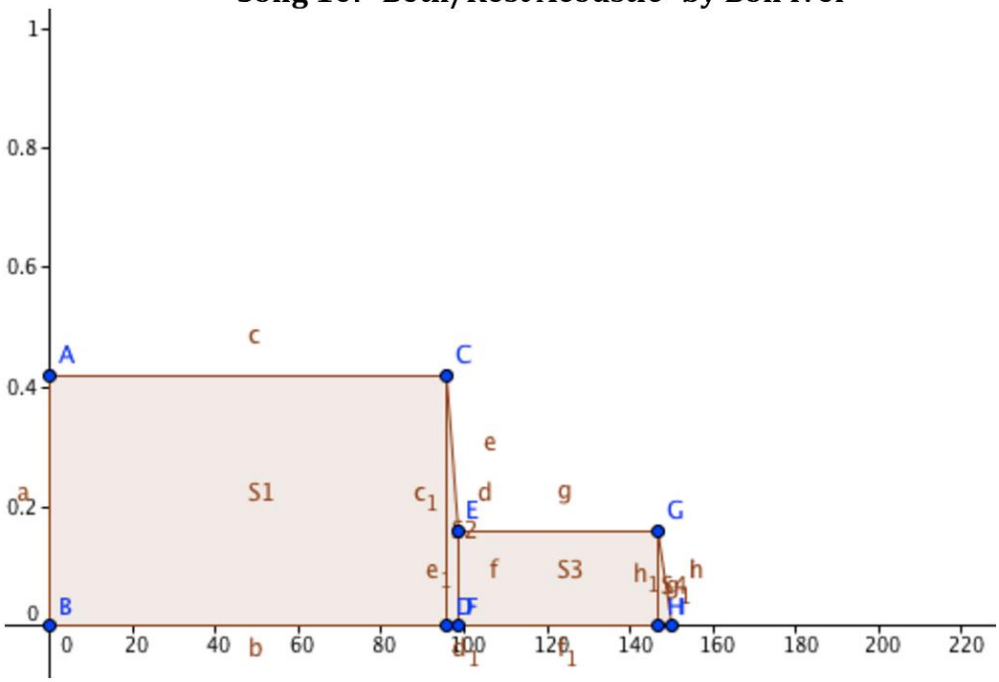

**Song 8: "The Fox" by Ylvis**



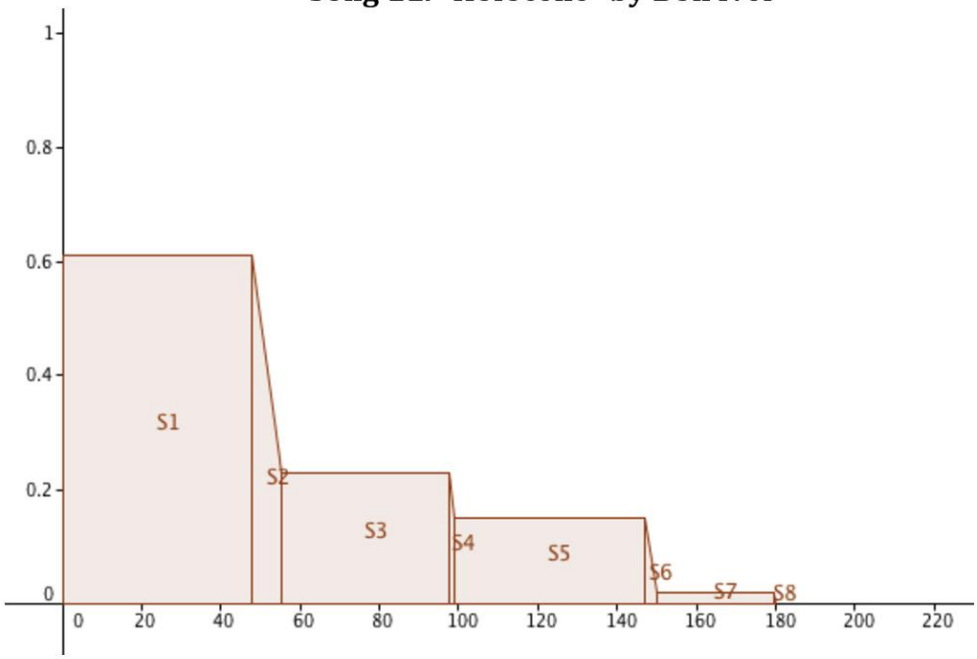
**Song 9: "The Cave" by Mumford and Sons**



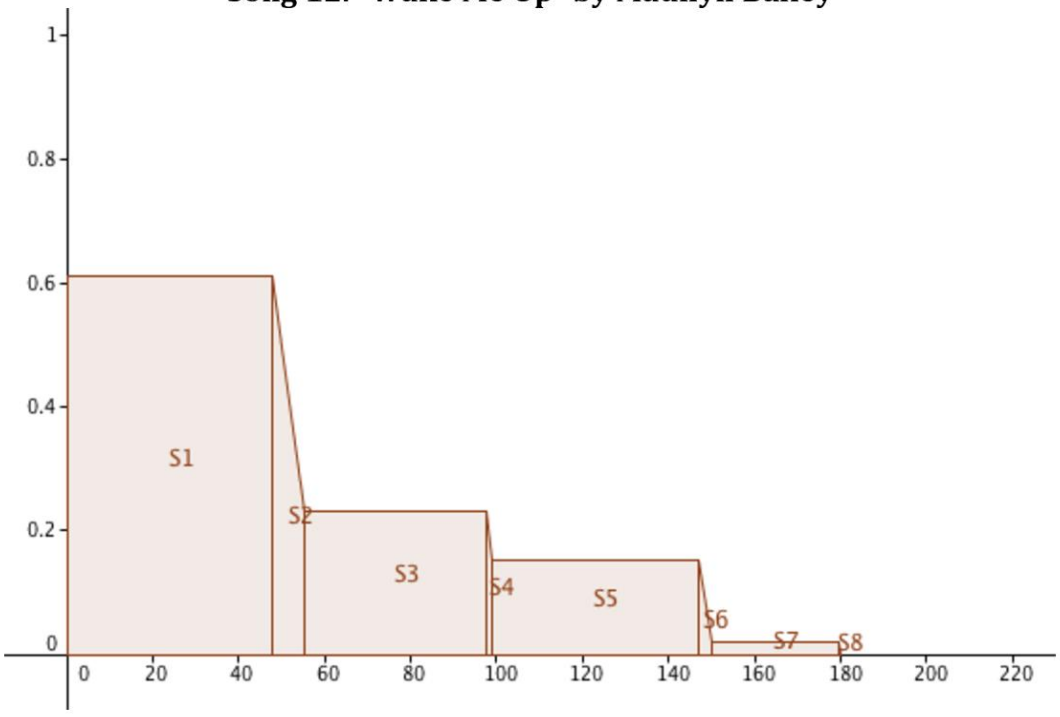

**Song 10: "Beth/Rest Acoustic" by Bon Iver**



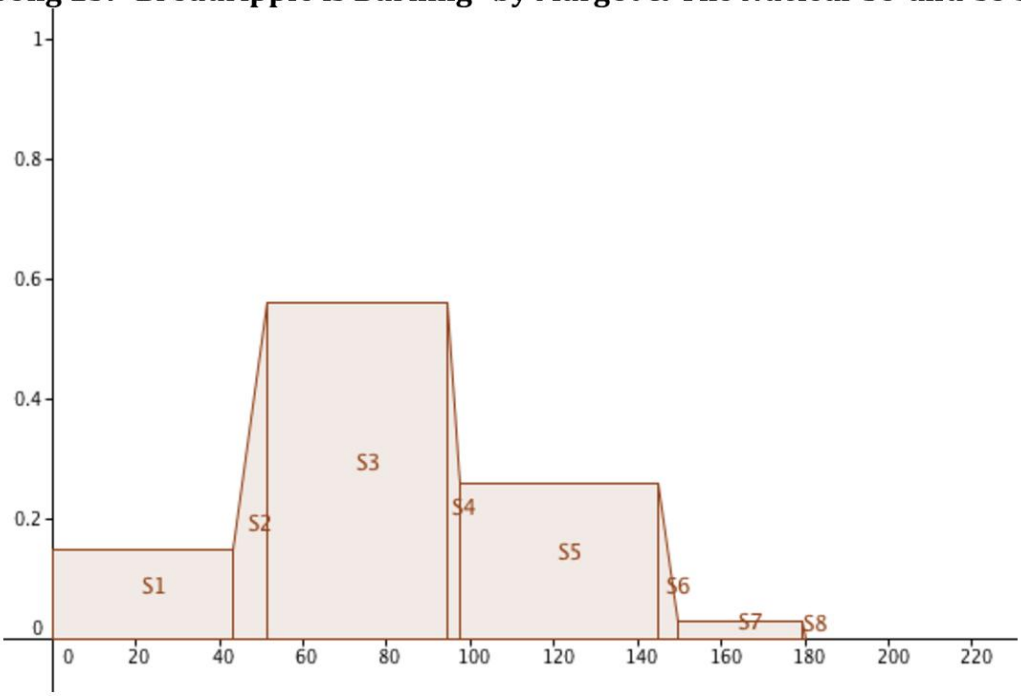
**Song 11: "Holocene" by Bon Iver**



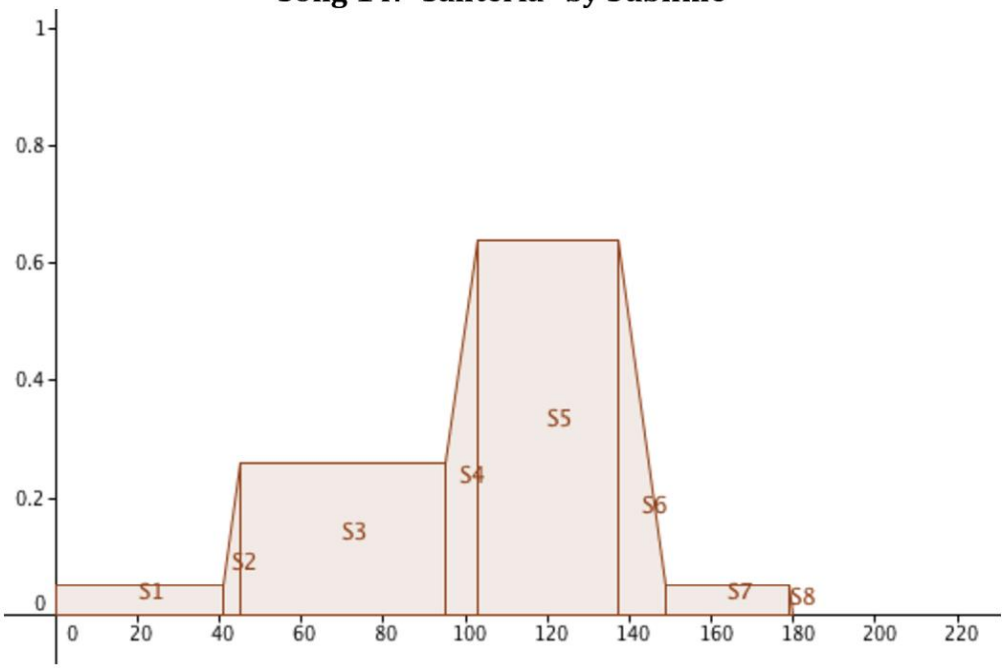

**Song 12: "Wake Me Up" by Madilyn Bailey**



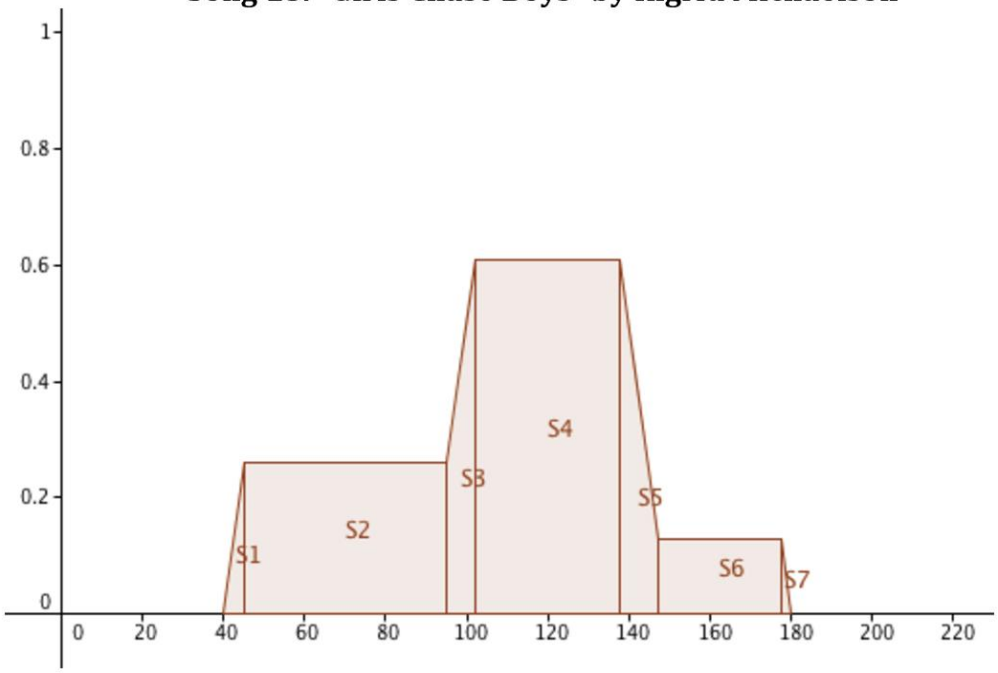
**Song 13: "Broadripple is Burning" by Margot & The Nuclear So-and-so's**



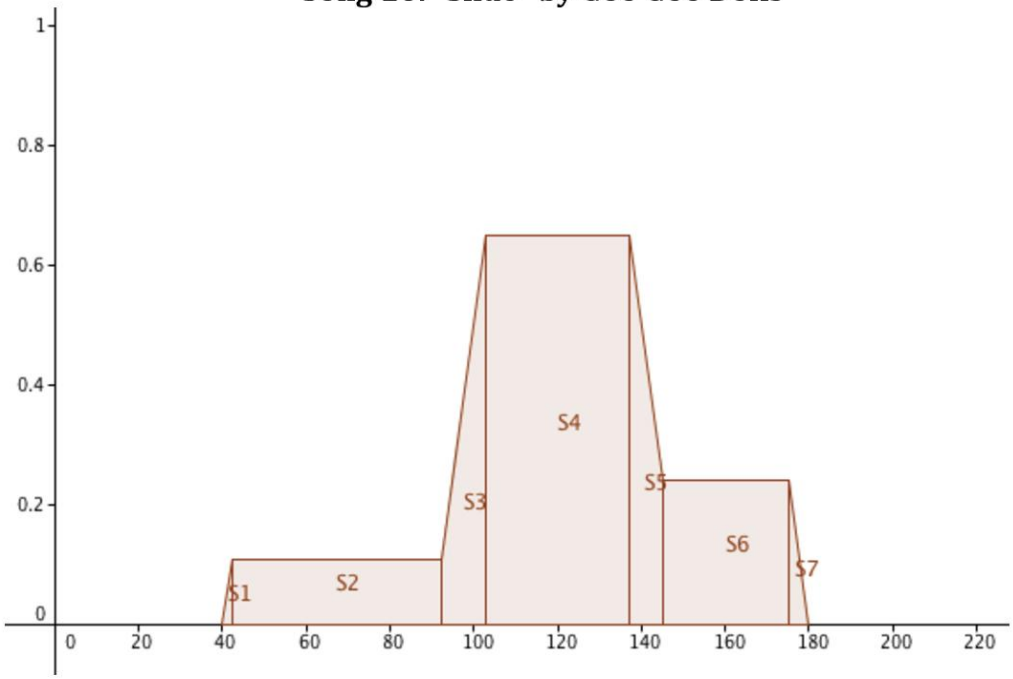

**Song 14: "Santeria" by Sublime**



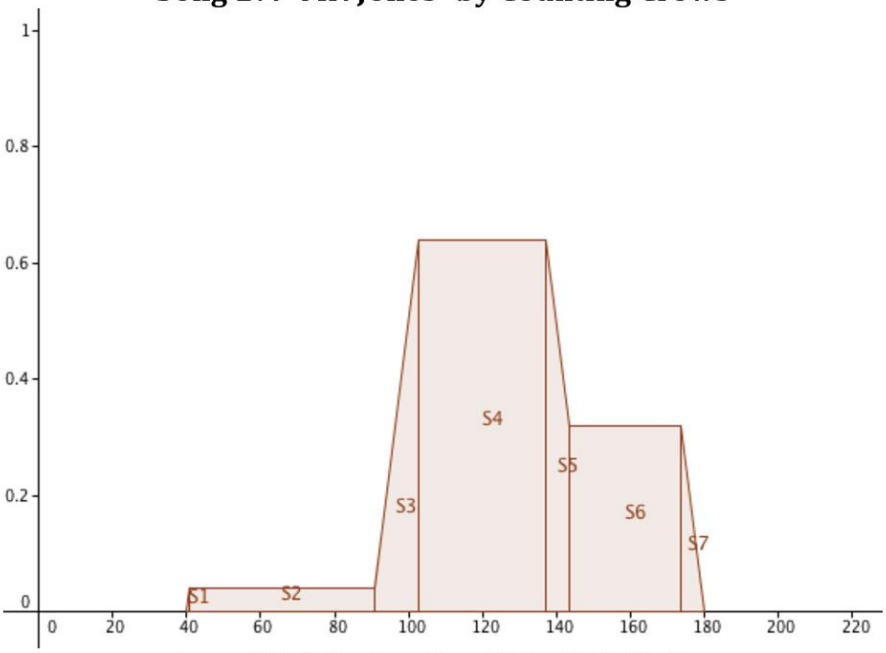
**Song 15: "Girls Chase Boys" by Ingrid Michaelson**



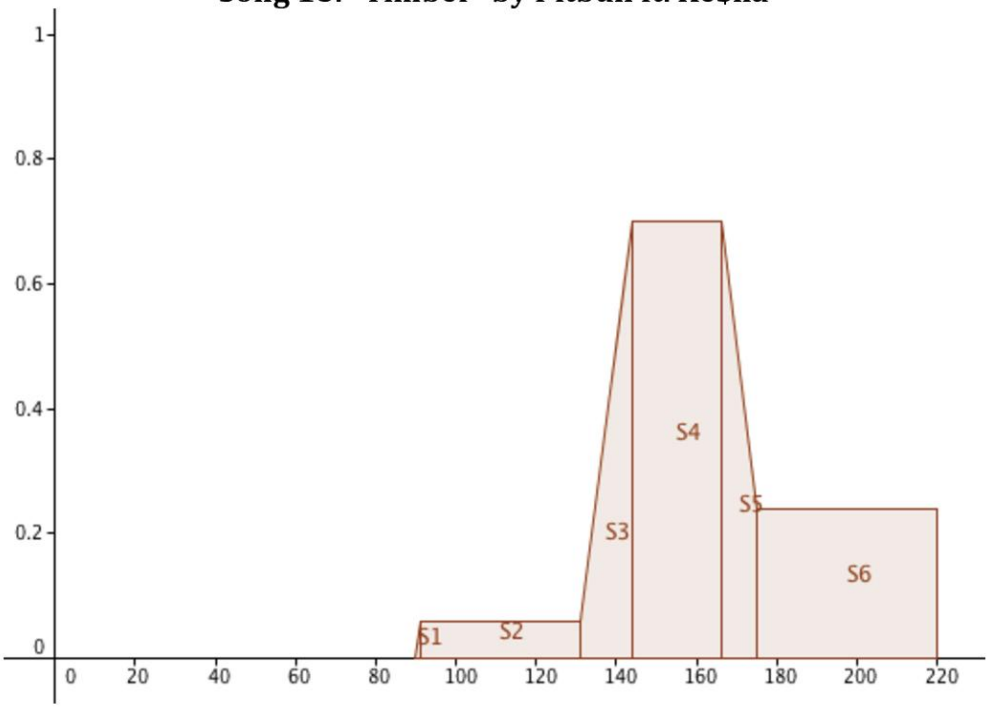

**Song 16: "Slide" by Goo Goo Dolls**



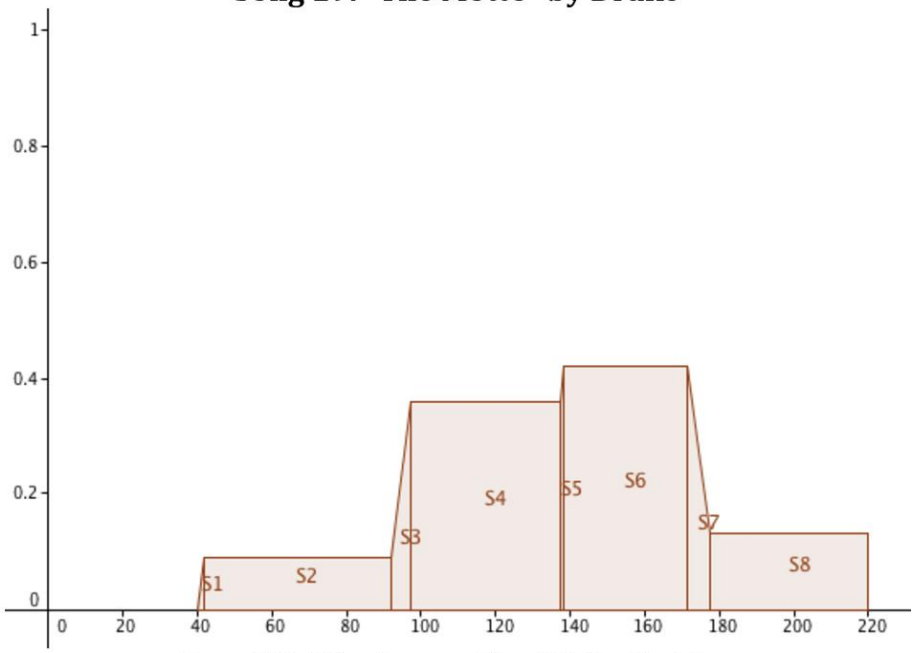
**Song 17: "Mr. Jones" by Counting Crows**



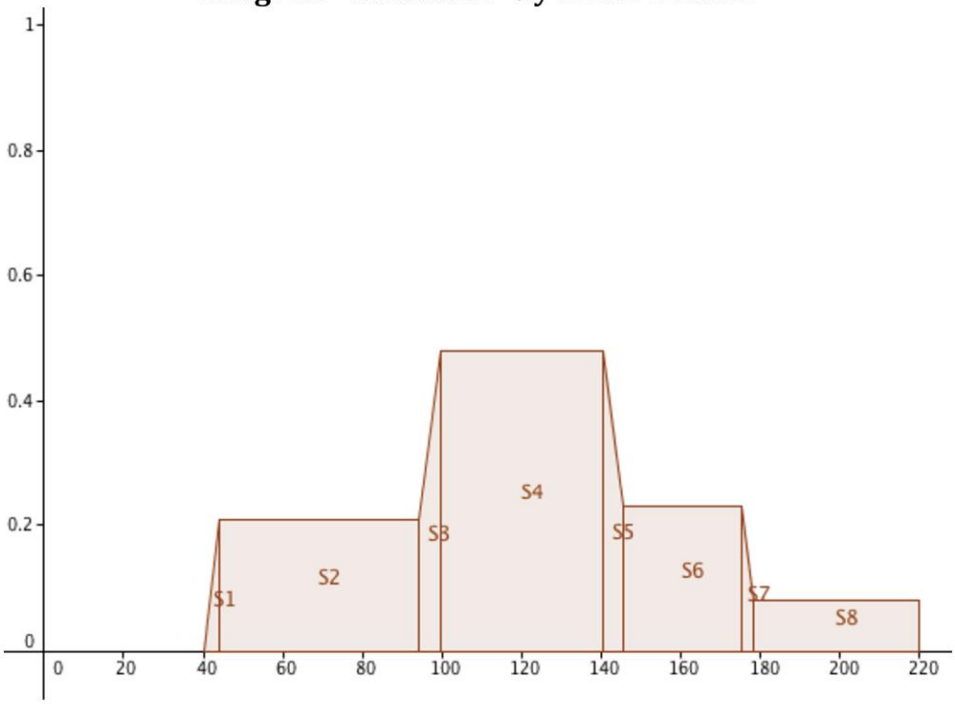

**Song 18: "Timber" by Pitbull ft. Ke$ha**



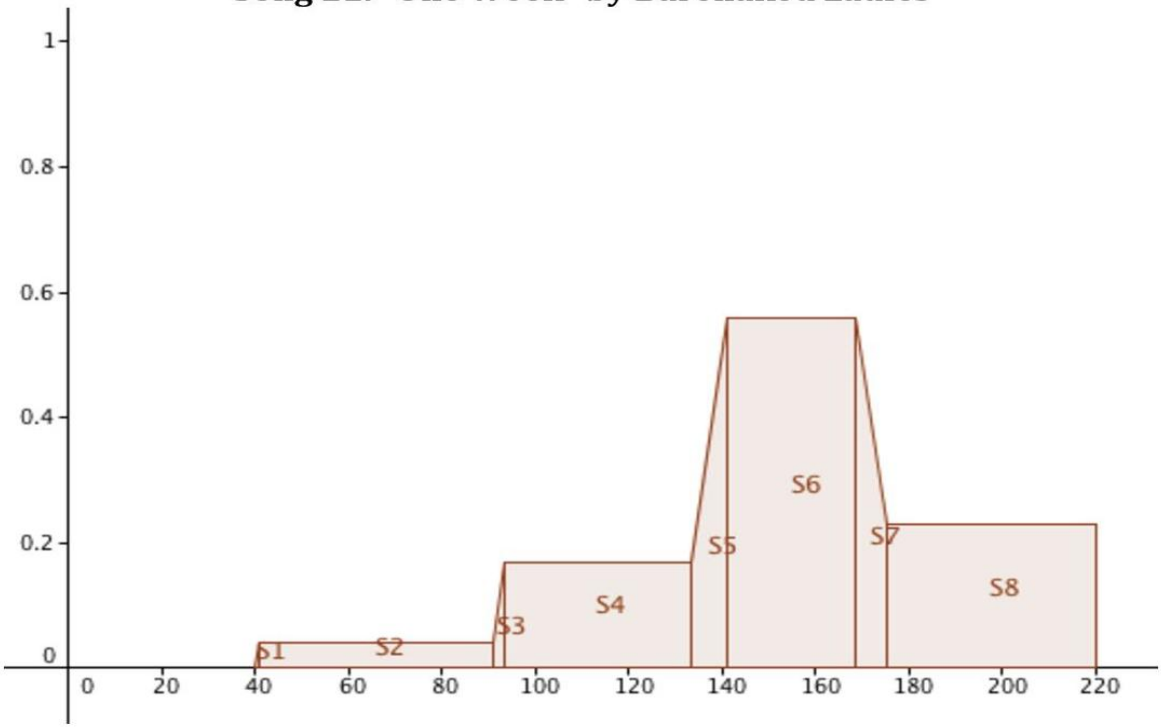
**Song 19: "The Motto" by Drake**




**Song 20: "Tuolumne" by Eddie Vedder**



**Song 21: "One Week" by Barenaked Ladies**



## 9.3 Results of FIS

Below is the list of songs with the PS, standing for perceived speed, calculated by the FIS.

| Song # | Data Type | Song Name | Artist | PS |
|---|---|---|---|---|
| 1 | training data | On Top of the World | Imagine Dragons | 127.8 |
| 2 | | I Love It | Icona Pop | 156.6 |
| 3 | | Entertainment | Phoenix | 149.5 |
| 4 | | The Real Slim Shady | Eminem | 131.2 |
| 5 | | I'm Gonna Be (500 Miles) | The Proclaimers | 108.4 |
| 6 | | Jubel | Klingkade | 119.3 |
| 7 | | Run the World (Girls) | Beyonce | 147.8 |
| 8 | | The Fox | Ylvis | 147.8 |
| 9 | | The Cave | Mumford and Sons | 100.8 |
| 10 | | Beth/Rest Acoustic | Bon Iver | 60.9 |
| 11 | | Holocene | Bon Iver | 50.4 |
| 12 | | Wake Me Up | Madilyn Bailey | 88.9 |
| 13 | | Broadripple is Burning | Margot | 73.4 |
| 14 | | Santeria | Sublime | 96.8 |
| | | | | |
| 15 | validation data | Girls Chase Boys | Ingrid Michaelson | 108.8 |
| 16 | | Slide | Goo Goo Dolls | 120.2 |
| 17 | | Mr Jones | Counting Crows | 126.9 |
| 18 | | Timber | Pitbull/Kesha | 163.9 |
| 19 | | The Motto | Drake | 128.5 |
| 20 | | Tuolumne | Eddie Vedder | 119.9 |
| 21 | | One Week | Barenaked Ladies | 153.7 |

## 9.4 Results of Neural Network Training

| **Song #** | **AS** | **PS** | **xr(1)** | **xr(2)** | **Rules** | **yr(1)** | **yr(2)** | **yr(3)** | **NN** | **=** |
|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 85.1 | 108.8 | 1.9299 | 2.0366 | A | 0.1 | 0.9 | 0.1 | A | yes |
| 16 | 115 | 120.2 | 2.0607 | 2.0799 | A | 0.1 | 0.9 | 0.1 | S | no |
| 17 | 70.8 | 126.9 | 1.8500 | 2.1035 | A | 0.1 | 0.9 | 0.1 | A | yes |
| 18 | 130.2 | 163.9 | 2.1146 | 2.2146 | F | 0.1 | 0.1 | 0.9 | F | yes |
| 19 | 133 | 128.5 | 2.1239 | 2.1089 | A | 0.1 | 0.9 | 0.1 | S | no |
| 20 | 85.8 | 119.9 | 1.9335 | 2.0788 | A | 0.1 | 0.9 | 0.1 | S | no |
| 21 | 56.4 | 153.7 | 1.7513 | 2.1867 | A | 0.1 | 0.9 | 0.1 | F | no |