Measuring caloric intake using chewing sounds

By

Dong Gee Hong

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Biomedical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2013

Date of final oral examination: 06/05/13

The dissertation is approved by the following members of the Final Oral Committee:
Willis J. Tompkins, Professor, Biomedical Engineering
John G. Webster, Professor Emeritus, Biomedical Engineering
Dale A. Schoeller, Professor, Nutritional Sciences
Yu H. Hu, Professor, Electrical Engineering
Walter F. Block, Professor, Biomedical Engineering

ABSTRACT

The main causes of obesity are related to diet. Two important factors in diet intake research are 1) identifying food types and 2) estimating consumed food mass. For this research study, I proposed a new method to identify food characteristics such as crispy, soft and mixed types by measuring sound patterns during mastication and recording using an ear canal microphone and smartphone. For this pilot test, ten solid foods and three beverages were selected. The food was categorized into three texture groups: crispy, soft and mixed. All subjects were served 3-4 bites of each solid food and 1/3 cup of beverage. The first step was dividing raw sound data into several mastication (chewing) event sounds ending by a swallowing sound. Each food bite was further separated into single chews. Three feature vectors were computed from one mastication envelope: changes in chewing energy, duration of subsequent chews and variation of energy within a mastication. Five other different feature vectors, the Mel-Frequency Cepstral Coefficients (MFCC), centroid frequency etc. were calculated. K-Nearest Neighbors, Neural Network and Support Vector Machine were used to classify the food with a hybrid hierarchy structure (decision tree). The root node classified the samples into three texture classes. At the child node, it classified 10 foods. Direct classification by individual food item showed poor (22%) accuracy. However, at the root node (80%) and at the child node (45%) using hybrid classification showed better accuracy. In conclusion, changing patterns of the three feature vectors were useful to identify food with different textures. A major limitation for identification of particular foods was individual variance across different subjects. The variance using the same subject and same food were located in a tolerable range for class identification, sounds features from different subjects sometimes fell outside of the tolerable range. Using reference vectors for the same

subject increased the identification accuracy. The smartphone works well for monitoring daily activities. I suggest the need for adding more sensors to acquire different signals to improve future accuracy.

ACKNOWLEDGEMENT

First, I would like to thank my advisor Prof. John Webster. This work would not have been possible without his guidance and support. I also thank the rest of my committee: Prof. Willis Tompkins, Prof. Dale Schoeller, Prof. Yu Hen Hu and Prof. Walter Block for their helpful advice and guidance throughout this work.

I also thank Prof. Franco Cerrina, Prof. Lloyd Smith, Prof. Michael Sussman, Prof. Aseem Ansari and Dr. Michael Shorteed. Specially, Prof. Franco Cerrian, he passed away in 2010, who gave me a chance to work in UW-Madison and to start my PhD again. I am deeply missed him. Dr. Michael Shorteed is a one of my best friend and mentor. I am truely appreciate his advice.

I missed Center for Nanotechnology and my colleagues. It was my first job in U.S. I've been working five years and learning about Maskless Array Synthesizer and related knowledge.

Finally, I am immensely fortunate to have a loving and supportive family. I thank my wife, SooNyun Jung. She endured more than 10 years of my study in U.S. and helped me to study. She always encourage me to follow my dreams. Last, I am truly grateful for my daughter Hyejee, my son Eunchan and my parents for their love and support.

TABLE OF CONTENTS

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

Nomenclature as used in this dissertation

Bite (Acquisition)

A single mouthful of a food as seized with teeth or taken with an eating utensil and entered into the mouth

Chewing (event)

Tearing and grinding food in the mouth while adding moisture (saliva)so it becomes soft enough to swallow. It is a series of chew events. It starts with a bite and ends at swallowing event. Same as mastication.

Mastication (event)

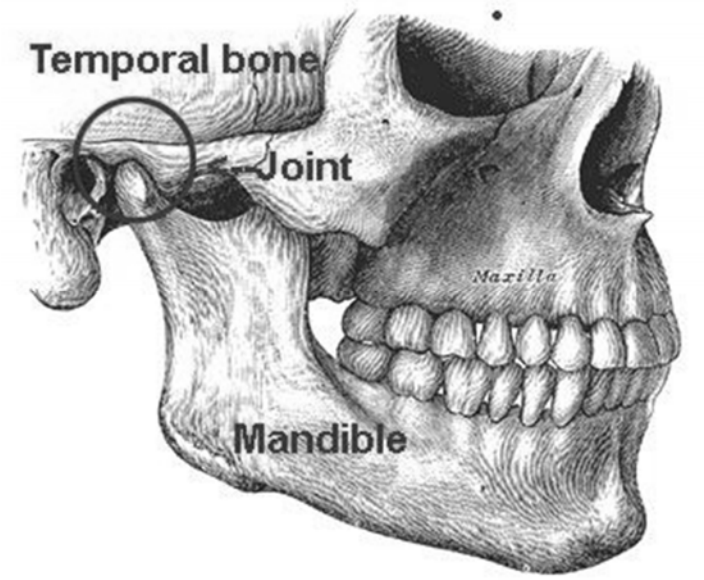
Tearing and grinding food in the mouth while adding moisture (saliva)so it becomes soft enough to swallow. It is a series of chew events. It starts bite and ends at swallowing event. Same as chewing.

Chew (event)

A single event to crush, grind, or gnaw (as food) with or with the teeth. It contains single act of opening and closing TMJ. The base unit of our research sound.

Temporomandibular joint (TMJ)

It is the joint of the jaw. *(Image is taken from eHow.com : http://www.ehow.com/way_5215257_tmj-cures.html)*



Crunching (Crackling, Crushing) sound

A noisy snapping or crispy sound from chew. Sound released from the breaking or rupturing of a firm structure. It is the characteristic sounds of a chew of a crispy food.

Squashing (Squishing, Squeezing) sound

A wet and soft sound from chew. It is the characteristic sounds of a chew of soft food.

Mouthful of food

The amount taken into the mouth at one time. It has same meaning of a spoonful or forkful of food in this research.

Sucking sound

A low power sound made when TMJ is opening, the soft, moist food sticks to the teeth then pull free allowing air into the subsequent void spaces.

Moisturize

Making wet or adding water content. The process of adding to create a moist paste in mouth. The food in the mouth turns to a paste during crushing, squashing and grinding.

One serving size

1 slice (of bread), the size of a deck of cards divided and placed side by side

1 cup, the size of a fist or a tennis ball

3 ounces cooked meat, the size of the palm of the hand or a deck of cards or a cassette tape

1/2 cup of potato chips, crackers or popcorn is one handful

Chapter 1

Introduction

1.1 Overweight and obesity

Obesity is one of the health problems in the United States. Obesity is common but it is serious and costly. In 2009–2010, more than one-third of U.S. adults (35.7%) were obese[1]. The percentage of children aged 6–19 years in the U.S. who were obese was 18% in 2010. During the last 30 years, it increased from 7% to 18% [1]. Overweight and obesity is more important for children because it affects their long term health and disease [35].

Overweight and obesity are problems because they can effect heart, lung and blood pressure and can cause obesity-related disease such as high blood pressure, diabetes, heart disease, high cholesterol levels, infertility, back pain, skin infections, ulcers and gallstones. These are top 10 obesity-related diseases.

Overweight and obesity are the result of “caloric imbalance” — too few calories expended for the number of calories consumed — and are affected by various genetic, behavioral, and environmental factors.

1.2 Statement of Problems

Considering the current approach to treatment of overweight and obesity, there are few approaches measuring caloric intake. On the contrary, there are many applications to control burning calories. The reason is that it is easy to measure.

One method of measuring caloric input is the patient diary. The method depends on the patient's memory. It is very simple but not accurate. The first reason is the reporting interval. If it is too short, they feel too busy, but if it is too long, the accuracy drops down quickly because it is hard to remember all the foods which they've eaten. So, I propose a new method to write down all food information they've eaten automatically. This approach is not new but it is still not well organized and developed. The previous research includes recording chewing sounds and other body electrical signals with a computer and analyzing them. However, they have problems. One is size of the recording system. The recording that we want to get has to be acquired in a real life environment, not in a lab. To carry a computer for 24 h will be difficult even if it is a laptop computer. Recent smart phone technology can put a computer into everyone's pocket. The development of computer engineering, most people carry a cellphone. In addition, the cellphone is changing into a smart phone. There are more than 133 million smart phones in the U.S. Most people don't have a problem carrying a smart phone. The new approach is to use a smart phone.

Another hurdle of recording is the patient's privacy. Generally, people won't reveal their status in public. So, the recording device can be hidden and not show outside. A smart phone based recording system is small enough to put in a bag or pocket and the microphone looks like a head set. These minimize the risk to expose the patients' status.

The other is the food type. Previous research focused on crispy texture, which has distinct features. But most foods have two or more ingredients and their textures are very different even if in single item. To deal with this issue, my approach deals with a variety of foods which have different textures.

This research will enhance a real caloric measurement application.

1.3 Proposed Approach

Several groups have begun to develop chewing and swallowing recording systems. Amft and Troster[6-8] reported the results of the first stage of their work on an automatic diet monitoring system. Their papers show detecting chewing sounds and swallowing sounds with recorded sounds and an EMG signal. They estimated bite weight by eight different parameters (Nr of chewing events, total chewing duration, mean event duration, variance of event duration, slope of event duration, chewing speed, slope of chewing speed and mean signal energy) with autoregression. One of the papers reported that they classified food by a clustering approach that has a hierarchical structure with wet-loud, dry-loud and soft-quiet types. The classifier showed 86.6% accuracy.

De Belie (2003) reports that chewing sounds of different types of dry-crisp snacks (two types of potato chips, prawn crackers, cornflakes and low calorie snacks from extruded starch) were analyzed to assess differences in sound emission patterns. The emitted sounds were recorded by a microphone placed over the ear canal. They used different multivariate analysis techniques for classification of the snack groups. The technique includes principal component analysis (PCA) and multiway data analysis. It showed 14 to 18% errors.

Nishimura et al.[9]made a wireless in-ear microphone with a Bluetooth protocol. The paper reports that chewing sounds of different types of crisp food (wafers, chips, apple, rice cracker, peanuts and salad) were analyzed to assess differences in sound emission patterns. The sounds were recorded by a wireless in-ear microphone. For feature extraction, they used Mel-Frequency Cepstrum Coefficients (MFCC). For training algorithm, they used Linde-Buzo-Gray (LBG).

Sazanov et al. [3]reported the automatic detection of swallowing events. They focused on ingestive behavior, not chewing sounds for different types of foods. They used a microphone to detect swallowing events. They proposed the wavelet packet decomposition method for feature extraction and SVM (Support Vector Machine) algorithm to detect the events. Its detection accuracy was 87.4%.

1.4 Dissertation Structure

Chapter 2 describes the experimental setup. It contains food selection, how to select subjects and details of the recording device. Chapter 3 shows basic properties of chewing signals. It contains segmentation which is the first step to process chewing sounds. I've compared the classical segmentation method, which is based on Otsu's algorithm and the new method that I've proposed. Chapter 4 shows several features of chewing sounds. These are divided into frequency domain and time domain. It includes Linear Predict Coefficients (LPCs), Mel-Frequency Cepstral Coefficients (MFCC) and other features to help classify data. Chapter 5 explains classification and identification from trained data. In this chapter, I've evaluated three algorithms such as K-Nearest Neighbor (KNN), Neural Network (NN) and Support Vector Machine (SVM). All algorithms have their identification rates. Chapter 6

shows analysis of bite weight estimation. I explained the experimental setup and the linear regression algorithm with data that I've evaluated. Chapter 7 contains a summary of the thesis and conclusions. It addition, I showed future directions for further study.

Chapter 2

Experimental Setup

2.1 Selection and preparation

Food selection is most important part of the research because the research focuses on eating habits in real life. In previous research, they selected food which has distinct texture properties such as apple, chips, lettuce, etc. But in this research, I've considered food textures and their popularity. For popularity, two food popularity surveys which contain fifty major contributors of calories in the US diet have been cited [11] [12].

	1976–1980	**2005–2006**
1	White bread	Grain based desserts
2	Doughnuts, cookies, cake	Yeast breads
3	Alcoholic beverages	Chicken
4	Whole milk	Soda/energy/sports drinks
5	Hamburgers, cheeseburgers	Pizza
6	Beef steaks	Alcoholic beverages
7	Regular soft drink	Pasta
8	Hot dogs, ham	Tortillas, burrito, tacos
9	Eggs	Beef
10	French fries	Dairy desserts
11	Cheese	Potato chips
12	Pork	Burgers
13	Ice cream	Reduced fat milk
14	Whole wheat	Cheese
15	Mayonnaise, salad dressing	Cereals

Table 2.1. *Top 15 sources of calories among Americans, NHANES 1980 and NHANES 2005-2006 [11] [12]*

From the survey, ten foods and three drinks were selected. The grayed cell are the food that I selected.

	Food	**Texture Type**
1	Whole wheat bread	Soft
2	Burger	Mixed
3	Carrot	Crispy
4	Cereals with milk	Mixed (Crispy)
5	Chicken	Soft
6	Potato chips	Crispy
7	Cookies	Crispy
8	Potato fries	Soft
9	Lettuce	Crispy
10	Pizza (Pepperoni)	Mixed
	Drink	**Type**
1	Water	
2	Soda	Carbonated
3	Milk	Viscosity

Table 2.2. *Selected food for the experiment and their chewing textures*

Table 2.2 shows, eight solid food were selected based on Table 2.1 and two were selected for crispy vegetables. The selection contains breads which contribute the most calories, popular meat such as chicken in the 2010 survey and potato chips, which is the most popular snack in US in the 2010 survey. There are two vegetables because those are the most popular ingredients for salads. In case of salad, although dressing contributes the most calories but it has to be removed because it is impossible to identify. For the drinks, three items were selected and they can cover most drinks.

Food texture is another consideration of the selection. They were organized by crispy, mixed and soft food since the research is based on chewing sounds. Mixed food includes food which is made with different ingredients such as pepperoni pizza, cheeseburger and cereal with milk. Especially, cereal with milk shows different textures with increasing time since milk changes microstructure of cereal. Lettuce and carrots were added for its texture, not calories.

Preparation of foods

Due to subject safety, all food which was used for the experiment was stored in the refrigerator or freezer. If food required cooking, it was prepared by thawing and cooking in the oven 10 min before the experiment.

For the amount of food, 1/8 pieces of pizza, ¼ pieces of cheeseburger and 40 g of chicken were prepared. 1/3 cup were prepared for drinks. For safety, leaving uneaten food was allowed in the experiment.

Knifes, folks and spoons were supplied for foods such as chicken and lettuce that required silverware to eat. All food preparation was done in the kitchen within the lab.

2.2 Setup

Subjects

All the experiment were approved by the IRB. The approval included number of subjects, location of the experiment, food types and storage of data to protect subject privacy.

Twelve subjects were approved and there were no records of sex, names and ages. All subject were volunteers. Especially, subjects who had a relation with the experimenter were eliminated.

From the volunteer list, two cases, food allergy such as lactose intolerance and TMJ dysfunction that makes jaw sounds while chewing, were eliminated.

For the recording device, the in-ear plugs were cleaned by alcohol swabs before the experiment. When the recording was finished, the signal data were stored in a secure network server which was protected by password and secure connection. All original recordings will be deleted in the server after finishing all analysis.

Experiment

Two categories of experiment were done. One was designed for identifying food type using chewing sounds and the other was estimating bite weight.

For identifying food type, twelve subjects consumed ten solid foods and three drinks. To easily distinguish, there were 30 s gaps between each food. During the gap, drinking water was allowed. Basically, each food required 3 swallows or more.

Quiet and noisy environments were made by radio for comparing signals for two different environment. For the 12 subjects, 9 were tested in the quiet environment and 3 were tested in the noisy environment.

For estimating bite weight, two subjects were involved. They had six plates of different amounts of potato chips which were 2 g, 4 g, 5 g, 8 g, 10 g and 12 g. It also had 30 s gaps between each plate. The amount was adjusted to not exceed the one day recommended sodium intake.

2.3 Recording device

The proposed method to record chewing sound was using a noise reduction in-ear microphone. Amft [7, 8] recorded chewing sounds in the ear but their in-ear microphone

permits some recording of environmental noise. Amft [8] show a gap between the ear canal and the microphone. Our ear microphone goes deeper into the ear canal and is surrounded by a silicone isolator. Our advantage is isolating the environmental noise while permitting normal hearing through the other ear. During preliminary studies, I investigated many microphones and selected and tested the WM-64K microphone. Its bandwidth is 50 Hz to 20 kHz.

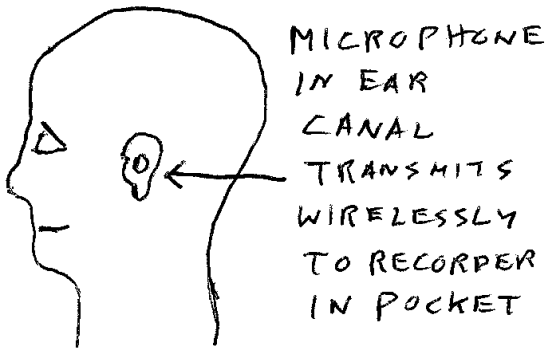


Figure 2.1. *A miniature microphone in the ear canal senses chewing sounds by bone conduction. It transmits for 24 h wirelessly to a recorder in the shirt pocket or waist pack.*

Our preliminary studies used an iPhone. It had an external microphone input jack and GPS. One device performed two functions at a time. For recording, the 'Recorder pro' application was used. For GPS logging the 'Trails' application was used. Most of the energy from chewing sounds is distributed from 50 Hz to 10 kHz. So we used a 22.1 kHz sampling rate. Data resolution was 16 bits. It required a 4.5 GB memory for 24 h recording. After attaching an external battery, total weight was 237 g (iPhone: 137 g, ext. battery: 100 g).

Figure 2.2. *NR microphone in the ear and attenuator at the end of the cable*

Figure 2.3. *External battery pack*

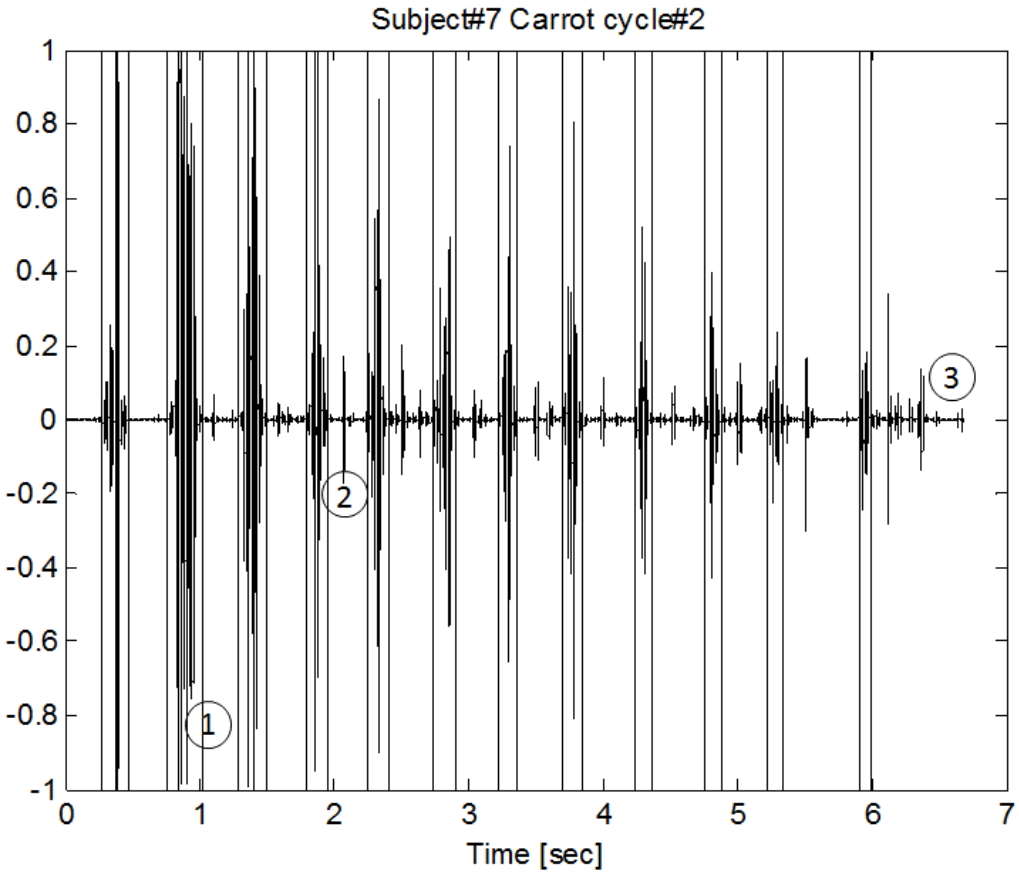
Figure 2.4. *Smartphone that is used for the research*

Chapter 3

Segmentation

3.1 Background

The following shows the basic property of chewing sounds.



① Chewing event - Bite

② Chewing event - Release

③ Swallowing event

Figure 3.1. *Basic definition of chewing sound by subject 7 with a carrot*

The basic cycle of chewing is based on three steps. First is a bite that chews food in mouth. Second is the release that finishes the first step. In this step, the jaws are open. Third is swallowing that swallows chewed food.

Figure 3.2 shows a cycle of chewing sound spectrum for high crisp food with single ingredient and low crisp with multiple ingredients foods. Generally, each chewing time is 200 ms to 350 ms but sometimes longer than that because individual eating habits are different.

For food textures, crispy food with a single ingredient shows a distinctive decreasing pattern as the number of chewing steps increase. On the contrary, multiple ingredient food shows random patterns during the number of chewing steps. This characteristic will be useful to distinguish two different food types, single or multiple ingredients. One of the reasons why the sizes of waveforms are random is that they contains crispy ingredients like pickles and crispy parts of a pizza.

The cycles of chewing sounds needed to extract individual chewing sounds for further signal processing. I extracted the peak size of the signal. I selected a threshold level and extracted the signal manually. Getting the power of the individual signal was the next step. For consistency, I removed the upper 10% of the high power signals because they can be in a transient state. For the same reason, I removed the lower 50% of individual signals because they can have different signal characteristics. They were a later part of the chewing cycle and were already moisturized. As described earlier, the duration of individual chewing cycle was 200 ms to 300 ms. If the duration was shorter than this, I also removed the signal.

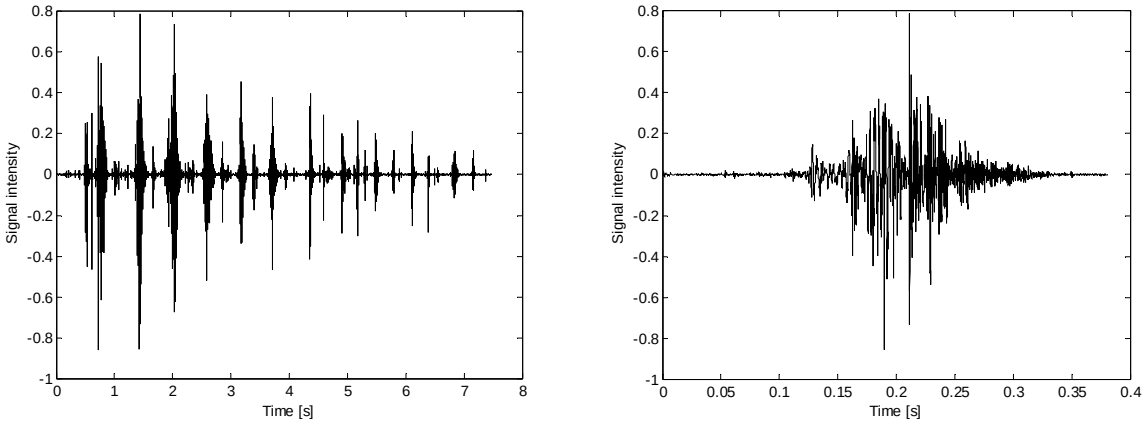


Figure 3.2. *A cycle of chewing sound spectrum for potato chips (left) and a sample of chewing sound (right).*

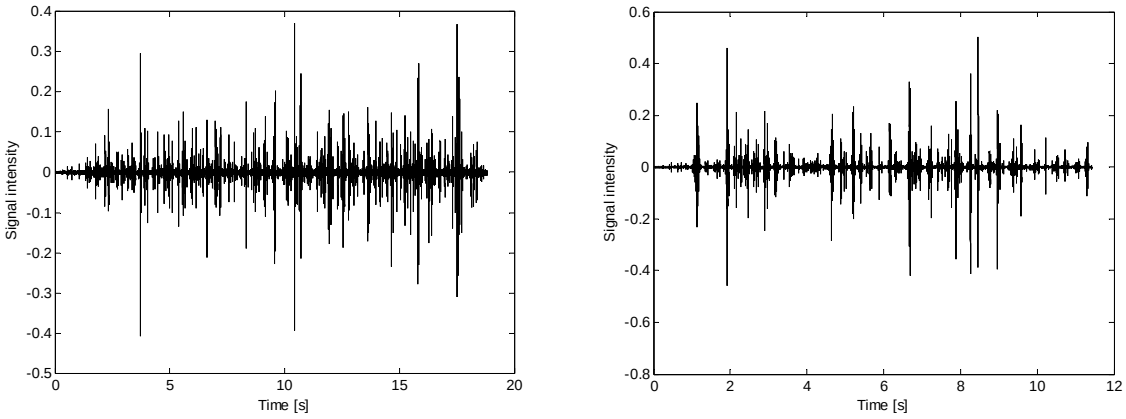


Figure 3.3. *Chewing sound spectra of cheeseburger (left) and pizza (right)for multiple ingredients food*

Texture based Crispy vs. Noncrispy

As expected, crispy food showed higher signal intensity (or energy) for earlier chewing events and the signal gradually decreased as chewing progressed because the food crushed, ground and finally changed to granules and wet.

In contrast, mixed and soft food showed a random intensity (or energy) pattern as chewing progressed.

Drink

A possibility was that while drinking carbonated drink (soda), bubble popping sound could be heard. Then it could be captured by the current in-ear microphone to detect drinking carbonated water.

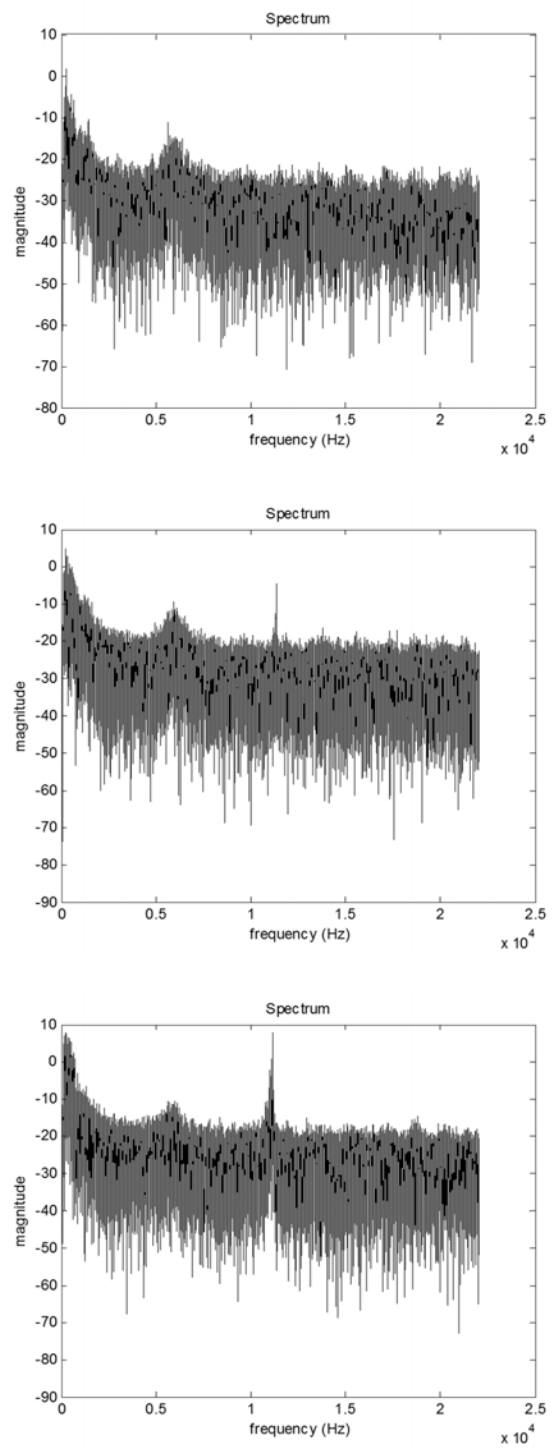


Figure 3.4 *power spectrum for three types of drinks in mouth*

Unfortunately, from the Figure 3.4 frequency analysis for the drinks, it was impossible to distinguish carbonated water among drinks. The reason was that the recording device could only record bone conduction sound. The bubble sound couldn't transfer by bone.

Signal processing flow for chewing sound

Figure 3.5 shows the basic processing step of analysis for chewing sounds.

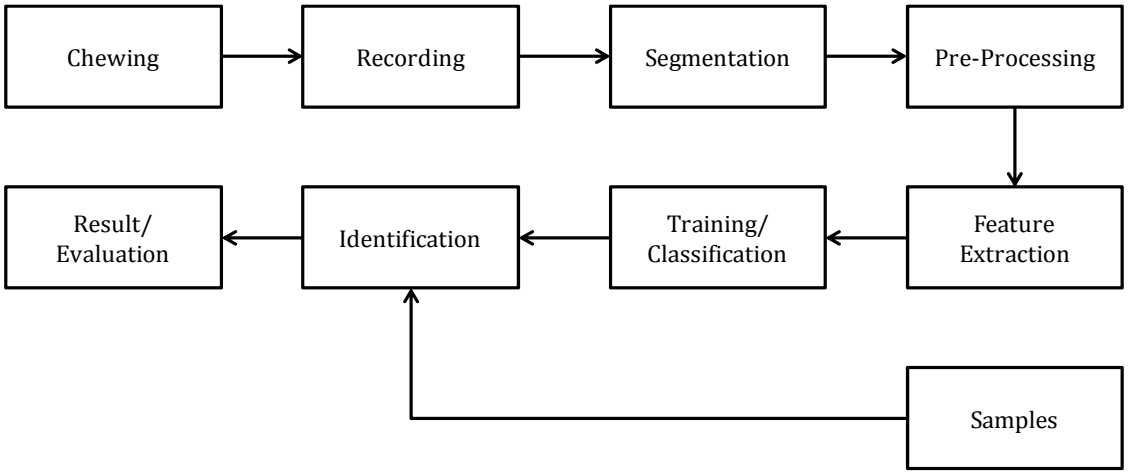


Figure 3.5. *Signal processing flow compares chewing sounds with stored samples.*

The swallowing sound makes up one cycle of chewing events. Eating one food means this cycle repeats several times and the basic issue was finding these cycles from the entire recording. After swallowing segmentation, to remove blank times, there was one more process for segmentation to split the signal part and nonsignal part.

Only signal parts were used for feature extraction. Feature extraction converted the time domain signal to parameters. The parameters were extracted by several ways such as LPC (Linear Prediction Coding), MFCC (Mel-Frequency Cepstral Coefficient) and other frequency parameters.

These parameters were classified and identified by K-NN (Nearest Neighbor), NN (Neural Network) and SVM (Support Vector Machine).

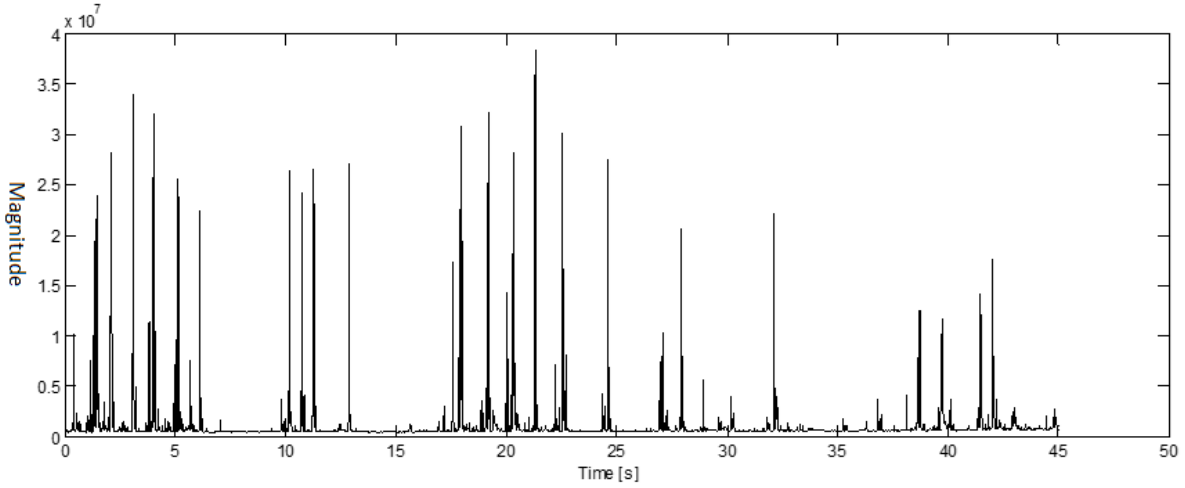
To estimate intake calories, two parameters had to be defined. One was food type and the other was bite weight. The second term was estimated by using the regression method.

There were two types of segmentation for the research. First was swallowing segmentation using the swallowing pattern.

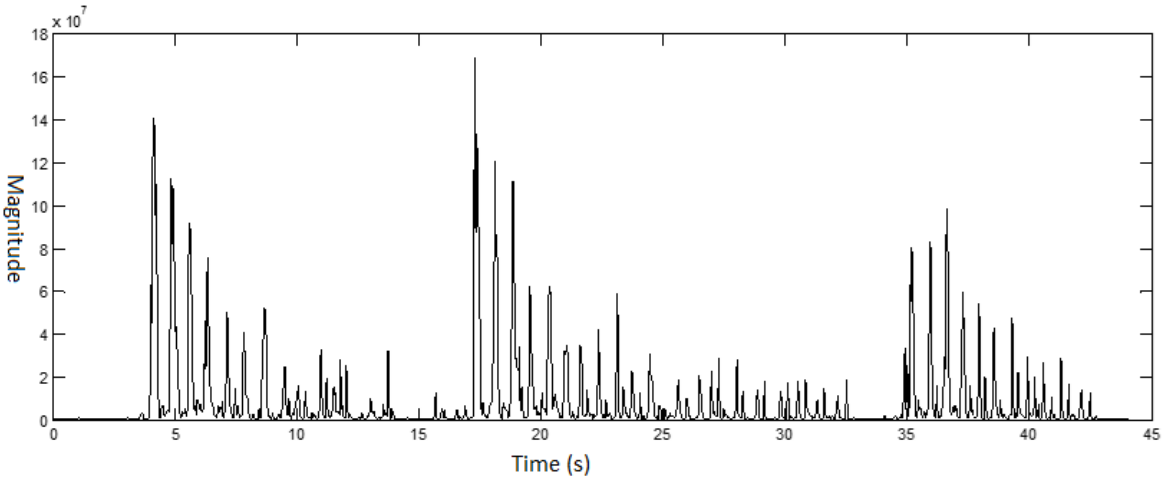
Swallowing segmentation for swallowing

1. Finding swallowing sound pattern from the original data. The average length of pattern was about 3500 samples.
2. Convert to frequency domain data
3. For every 3500 samples of original data, convert to frequency data. It uses 50% overlap.
4. Using Pearson's correlation
5. Repeat until it finishes

Figure 3.6 shows the data.



(a) Pearson's correlation for drinks



(b) Pearson's correlation for potato chips

Figure 3.6. (a) *Pearson's correlation for three samples of drink and (b) Pearson's correlation for potato chips with swallowing signal pattern*

In the experiment, in case of drink, the swallowing pattern was more than 98%. But in the case of solid food, it was impossible to find a pattern. For addition, it can use decreasing

trend. If the signal jumped and was large enough, the last small pattern was the swallowing pattern. But for this research, manual segmentation was used.

3.2 Classical algorithm

Classical segmentation splits signal data and nonsignal data. Different from other signals, chewing sounds had a special pattern that showed repeats of signal and nonsignal. The nonsignal part was low enough and the signal part was high enough. So the histogram based algorithm was well adapted for the segmentation. Most of the signal was in the low part and the peak values placed in the higher part. The lower threshold was generated from the histogram of signal intensity by selecting the 65% level of the signal intensity. However, to find the peak part, higher part data were useful.

Otsu's algorithm

The total number of individual chewing sounds was more than 2000. Increasing the number of subjects and types of food, the individual signals increase exponentially. To reduce the heavy load, suggests use of the automatic segmentation method. For the first part, I used the Otsu algorithm to get a threshold.

The Otsu algorithm uses a classification method. To get a threshold value, it uses a cost function and finds out its minimum. The best way to separate two classes is when both sides of the distribution are similar. This means the variance has to be small. So, the cost function is the sum of variance which has its own weight. The minimum value of the function is the threshold.

$$\sigma^2_{\text{Within}}(T) = n_{\text{B}}(T)\sigma^2_{\text{B}}(T) + n_{\text{O}}(T)\sigma^2_{\text{O}}(T)$$

where,

$$n_{\mathrm{B}}(T) = \sum_{\mathrm{i}=0}^{\mathrm{T}-1} p(\mathrm{i})$$

$$n_{\mathrm{O}}(T) = \sum_{\mathrm{i}=\mathrm{T}}^{\mathrm{N}-1} p(\mathrm{i})$$

$$\sigma_{\mathrm{B}}^2(T) = \text{the variance of the data below threshold}$$

$$\sigma_{\mathrm{B}}^2(T) = \text{the variance of the data above threshold}$$

and $[0, N-1]$ is the range of intensity levels. Computing this within-class variance for each of the two classes for each possible threshold involves a lot of computation, but there's an easier way. If you subtract the within-class variance from the total variance of the combined distribution, you get something called the between-class variance:

$$\sigma_{\mathrm{Between}}^2(T) = n_{\mathrm{B}}(T) n_{\mathrm{O}}(T) (\mu_{\mathrm{B}}(T) - \mu_{\mathrm{O}}(T))^2$$

So, for each potential threshold T

1. Separate the signal into two clusters according to the threshold.

2. Find the mean of each cluster.

3. Square the difference between the means.

4. Multiply by the number of signal in one cluster times the number in the other.

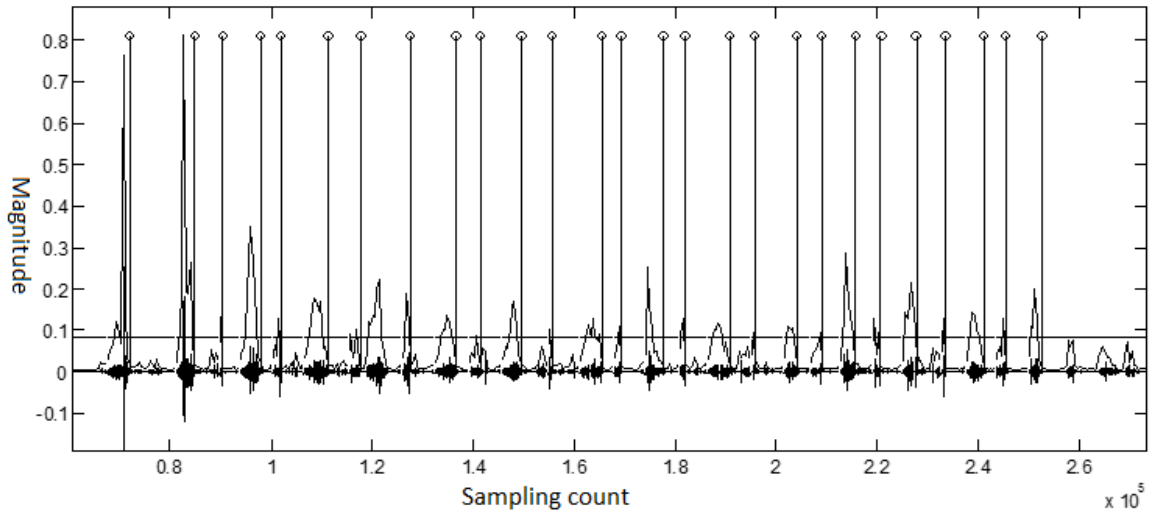


Figure 3.7. *Automatic segmentation using the Otsu method*

From the cycles of chewing sound, the threshold and local minimum forward of the point and backward were acquired. The signal between two points was an individual chewing sound. Also, if the duration was less than 200 ms, it was not selected.

10^6 individual chewing sounds were found from a whole wave file which had 115 peaks. The rate of extraction was 92%. Although the signal wasn't good, the upper 10% and lower 50% elimination helped to yield a better result.

3.3 New algorithm

The new algorithm uses a histogram.

1. Get the envelope of the chewing event signal

Envelope: Hilbert transformation -> Full-wave rectification -> Low pass filter -> Downsampling

2. Get histogram and compute the lower threshold and the higher threshold

3. Get peak and cross point in lower threshold from the envelope

4. Make a group from peak and two cross points

Getting the envelope from the original signal is the first step. For Otsu's algorithm, the moving average is used but in this algorithm it uses the Hilbert transform. Once it is computed, full-wave rectification, low pass filter and down sampling follows to continue the process.

The Histogram is used to compute the upper threshold and the lower threshold. The lower threshold is used to compute two points to determine the signal region. The upper threshold is used for finding peaks. The lower peak signal represents the release sound and the upper signal represents the bite signal. The release signal is similar although it comes from different foods. Thus the upper threshold is required to remove the release signal.

Otsu's algorithm distinguished 92% of bite signals and the new algorithm distinguished 98% of bite signals from the entire recording signal.

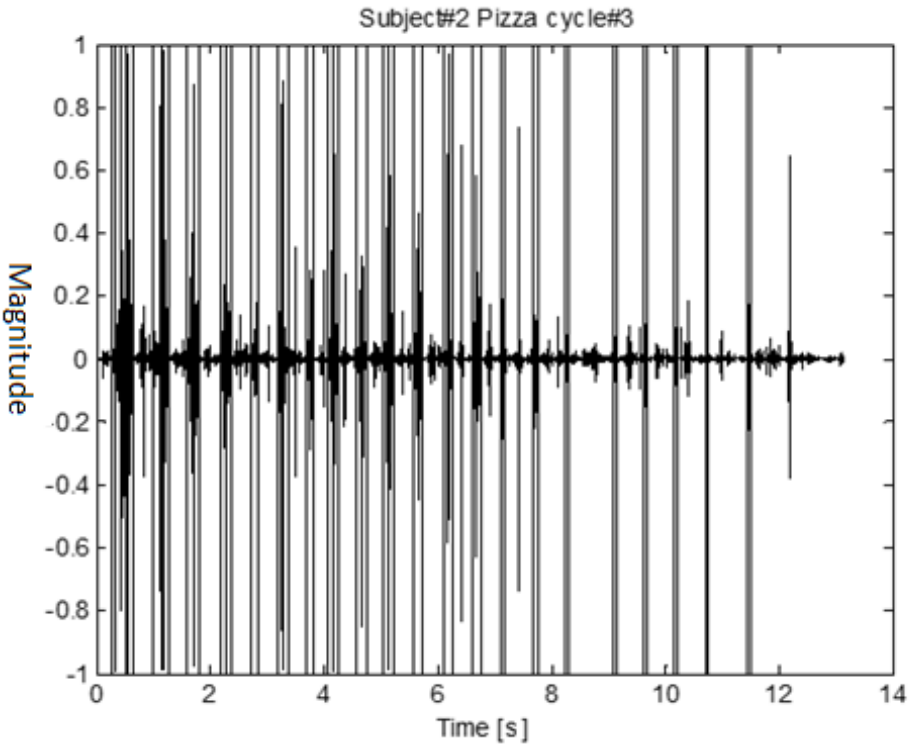


Figure 3.8. *Automatic segmentation using the newly proposed algorithm*

Chapter 4

Feature extraction

4.1 Frequency based features

There are several steps to process the chewing sounds. The important steps are signal sampling, feature extraction and recognition. The signal sampling first step is described above. The second step is feature extraction. It extracts certain features from the sounds. It generates parameters used by several methods for recognition. We used LPC (Linear Predictive Coding) [15], MFCC (Mel-Frequency Cepstral Coefficients) [20] These methods show differences among sound data.. Cepstrum is an inverse FFT of the log spectrum. It is widely used to analyze periodic signals because it shows the characteristics of periodic signals easily. MFCC makes the cepstral coefficients, which are used as predefined weight functions for different frequencies.

LPC (Linear Predictive Coding)

LPC is a method to extract features from speech signals. It assumes that the signal can be approximated as a linear combination of the past samples.

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n)$$

The linear combination of past samples as the estimated $\bar{s}(n)$

$$\bar{s}(n) = \sum_{k=1}^{p} a_k s(n-k)$$

The prediction error $e(n)$ is defined as

$$e(n) = s(n) - \bar{s}(n)$$

To minimize error $e(n)$, find the mean square.

$$\mathrm{E_n} = \sum_{\mathrm{m}} \mathrm{e_n^2}(m)$$

To solve this equation for the optimum predictor coefficients ($\hat{a}_k s$) we have to compute $\phi_{\mathrm{n}}(i,k)$ and then solve the resulting set of p simultaneous equations.

$$\phi_{\mathrm{n}}(i,0) = \sum_{\mathrm{k=1}}^{\mathrm{p}} \hat{\mathrm{a}}_{\mathrm{k}} \phi_{\mathrm{n}}(i,k)$$

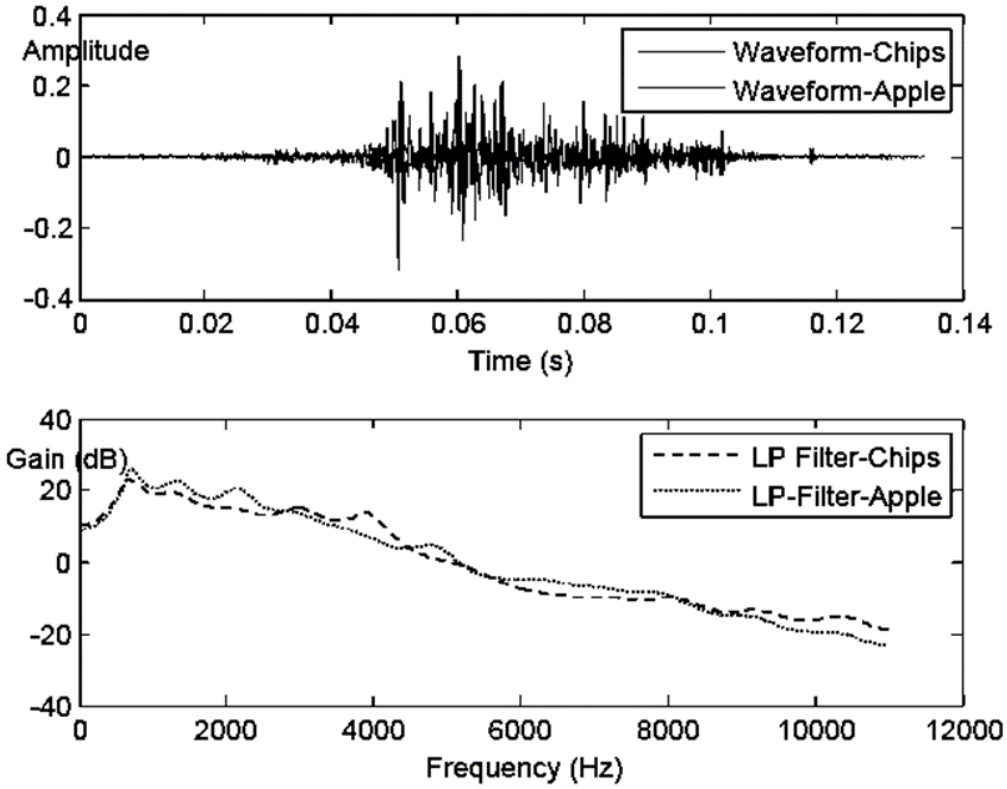


Figure 4.1. *Top: Single bite sound in the time domain. Bottom: Linear predictive coding frequency spectrum*

Fig. 4.1 top shows a single jaw closure sound in the time domain. Figure 4.1 bottom shows LPC spectral envelope (13 coefficients) in the frequency domain. It is a simple all-pole spectrum modeling. The shape of the LPC envelope for eating an apple and chips are similar. The correlation coefficient between the two data varies from 0.99 to 0.95. In Fig. 6 bottom, the correlation coefficient is 0.98 and it makes it hard to distinguish between apple and chips.

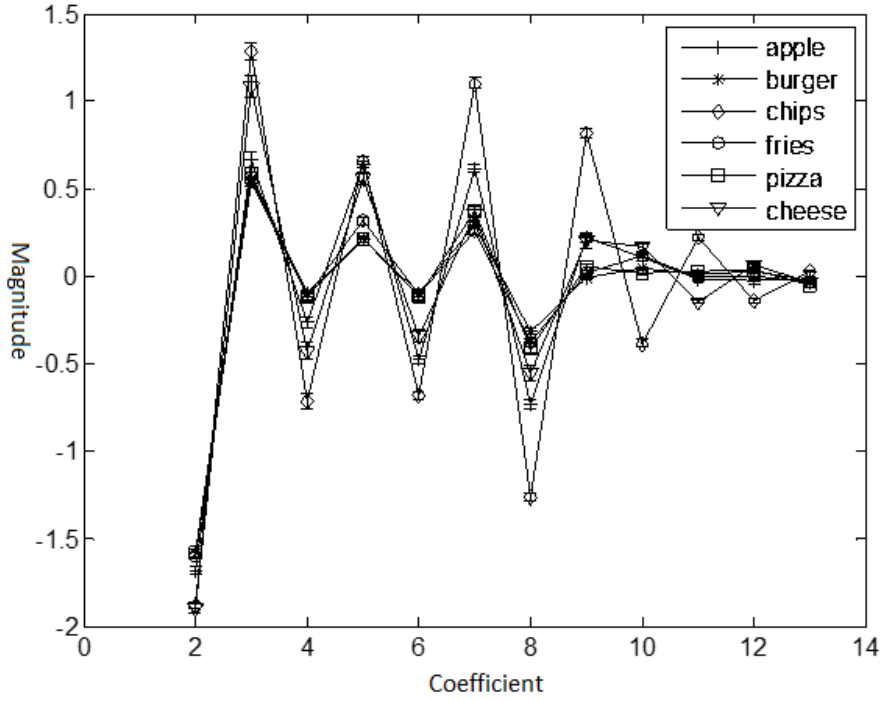


Figure 4.2. *Coefficients of LPC for various foods*

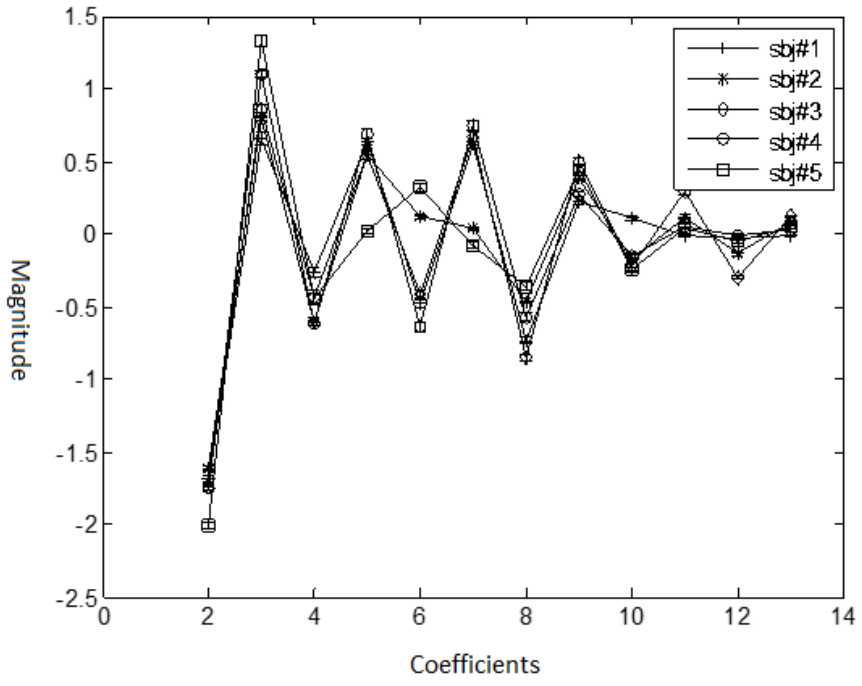


Figure 4.3. *Coefficients of LPC for different subjects*

There are two types of coefficient arrangement. One shows different foods for the same subject and the other shows the same food for different subjects. The main issue is the same food for different subjects.

Fig. 4.3 shows the result. Although they are chewing same food, they are making different patterned sounds. Subject 2 and subject 5 show different patterns than the others. It is important for further research because if the patterns are the same, then a small numbers of references is enough to classify them. But from my result, there were different pattern signals, so a large number of references or some numbers of group references were needed. It also yields less accuracy for food identification.

To compare foods, pizza and burgers have similar patterns. So, we can estimate the accuracy for identification is high except for pizza and burger signals.

MFCC (Mel Frequency Cepstral Coefficients)

MFCC is also a method to extract features from sound signals. The way to get the coefficients is shown in Fig. 4.4.

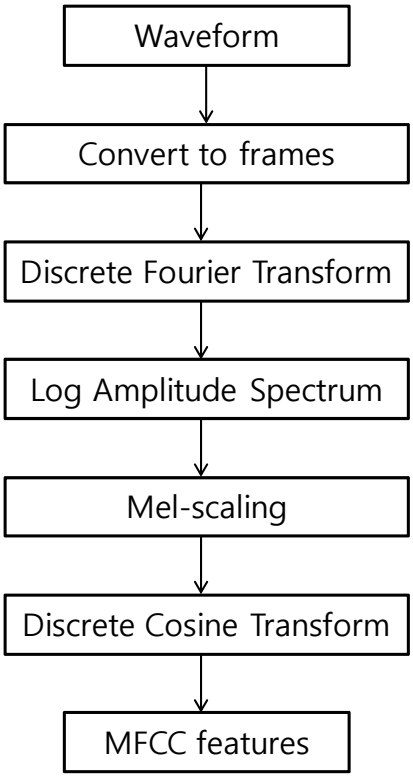


Figure 4.4. *Process to create MFCC features*

The Mel scale function is

$$\mathrm{m} = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

The Mel-scale is based on a mapping between the actual frequency and perceived pitch as apparently the human auditory system does not perceive pitch in a linear manner. The mapping is approximately linear below 1 kHz and logarithmic above 1 kHz.

A triangle filter is made by the mel-scale. The signal passes through the filter and then the filter separates it by a defined frequency range. Fig. 4.5 shows the mel-scaled triangle filter.

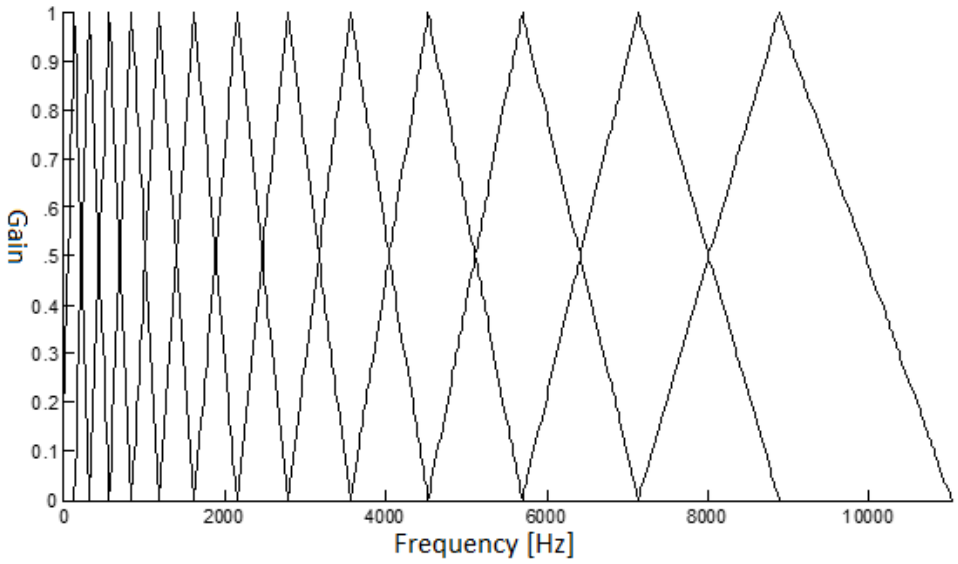


Figure 4.5. *Triangle filter with Mel-scale*

In the same manner, MFCC was applied for all sampled individual chewing sounds. MFCC makes various coefficients for different foods and different subjects. For the different foods,

it can have fewer distinguished coefficients to compare with coefficients of LPC. But for the different subjects, the differences are not large.

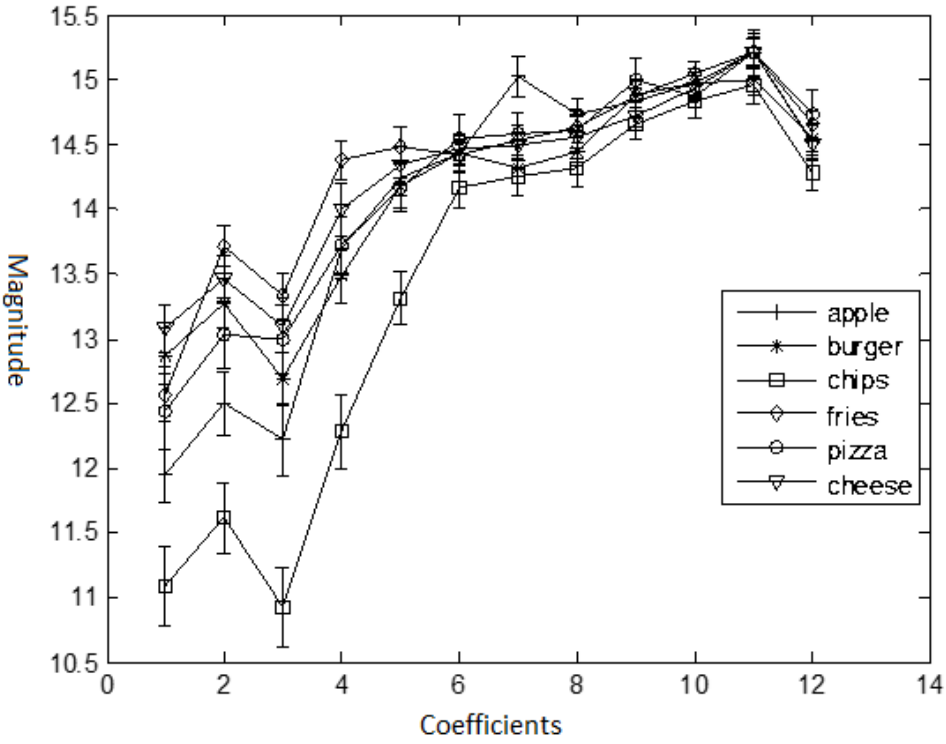


Figure 4.6. *Coefficients of MFCC for various foods*

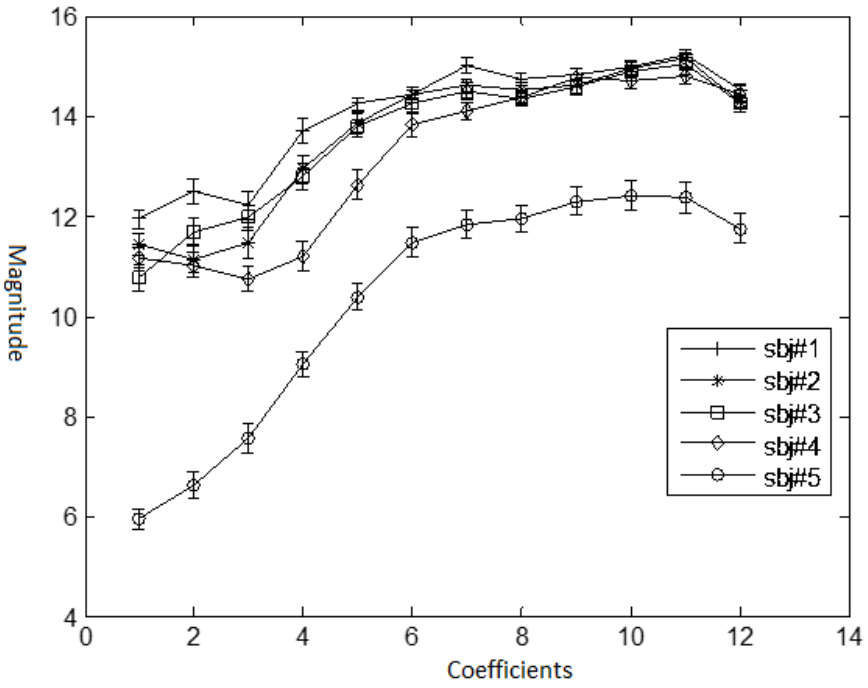


Figure 4.7. *Coefficients of MFCC for different subjects*

For both methods, there are smaller differences for higher coefficients. The lower coefficients are good enough to classify them.

Frequency centroid, bandwidth, sub-band power, power, and pitch are also useful methods to find differences in the frequency domain.

Frequency centroid

Frequency centroid represents center of energy distribution. It shows where the energy is focused. It can be used to represent the brightness of the sound.

$$F_C = \frac{\int_0^{f_o} f_i \times p_i \, df}{\int_0^{f_o} p_i \, df} \quad \text{where} \quad p_i = \log\left(\int_{L_i}^{H_i} |F(f)|^2 \, df\right), \; f_o = {}^1/_{f_s}$$

f_o is half of sampling frequency and f_i is arbitrary frequency.

Bandwidth

Bandwidth shows whether the spectrum is focused on the center frequency or distributed over all frequencies.

$$F_B = \sqrt{\frac{\int_0^{f_o}(f_i - F_c)^2 p_i df}{\int_0^{f_o} p_i df}}$$

Sub-band power

Sub-band power is the power spectrum in between the following sub-band frequency. The sub-band can be determined by signal features.

Band : $[0, 1/8f_o]$, $[1/8f_o , 1/4f_o]$, $[1/4f_o , 1/2f_o]$

$$p_i = \log\left(\int_{L_i}^{H_i} |F(f)|^2 df\right)$$

Power

Power includes the power spectrum for all frequencies.

$$P = \log\left(\int_0^{f_o} |F(f)|^2 df\right)$$

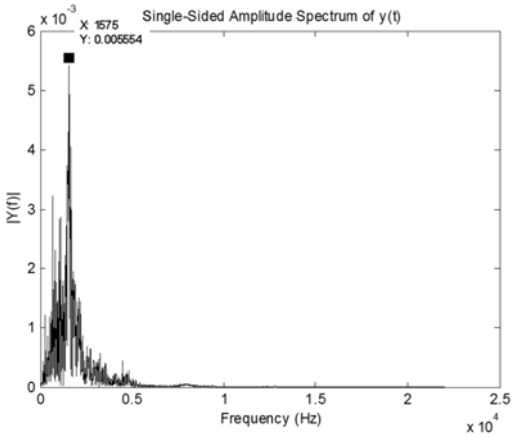
Pitch

Pitch can be calculated by autocorrelation and peak detection using all frequencies

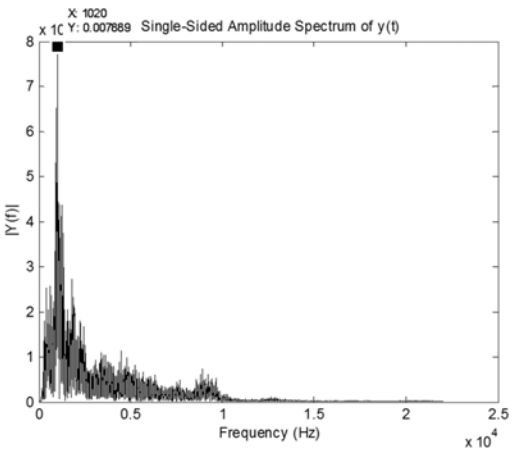
One mouthful of food yields several chewing events before swallowing. The number of chewing events depends on the subject's eating habits, food textures, and quantity of the food for one mouthful. The segmented data from the segmentation algorithm contain all chewing event signals from start to end. The results of the previous classification/identification use all segmented data. Explicitly, as the number of chewing events increases, the texture also changes. Changing texture means changing frequency features such as pitch, variations, etc. To improve performance for classification, the change in chewing events within one cycle of swallowing has to be considered.

Crispy texture

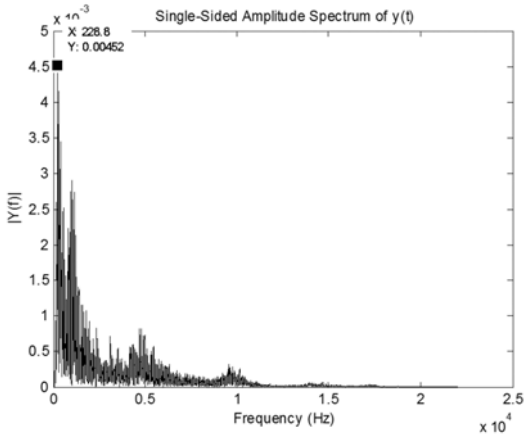
The first several events showedthat the chewing cycle crushed crispy food to wet it and make it less crispy. After the chewing events, the sound pattern became similar to that of soft food because it changed the food into small wet pieces. Figure 4.8 shows the sound pattern changes when chewing events increase. The peak frequency moves toward lower frequency as the texture changes to noncrispy.



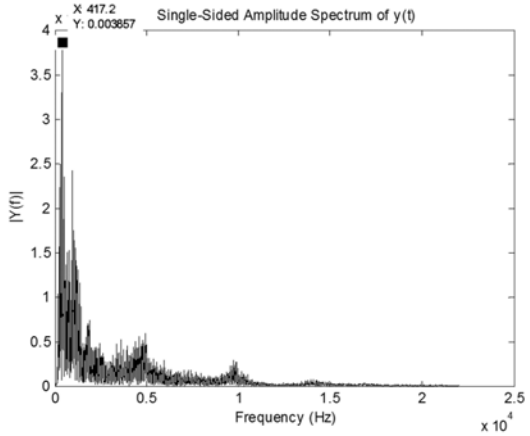
2nd bite, Pitch : 1575 Hz



3rd bite, Pitch : 1020 Hz



10th bite, Pitch : 228 Hz

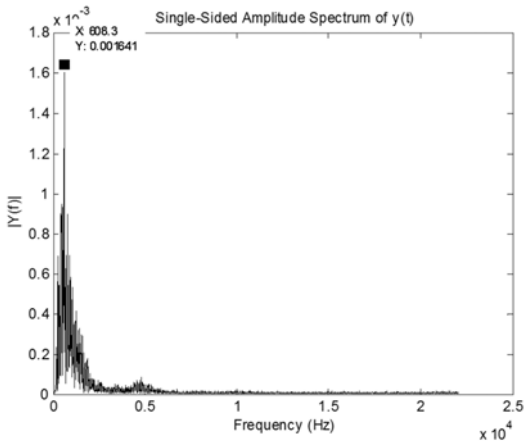


15th bite, Pitch : 417 Hz

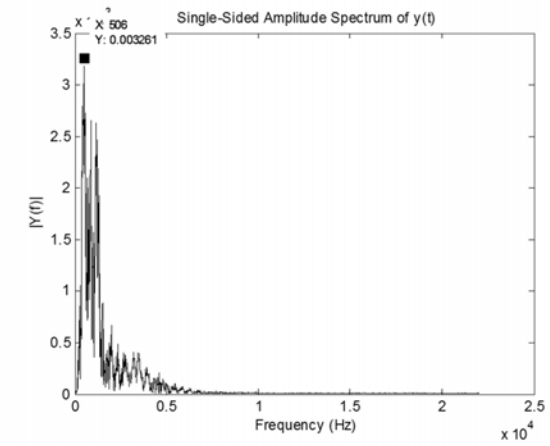
Figure 4.8. *Power spectrum of lettuce chewing sounds with arbitrary chewing*

Soft food

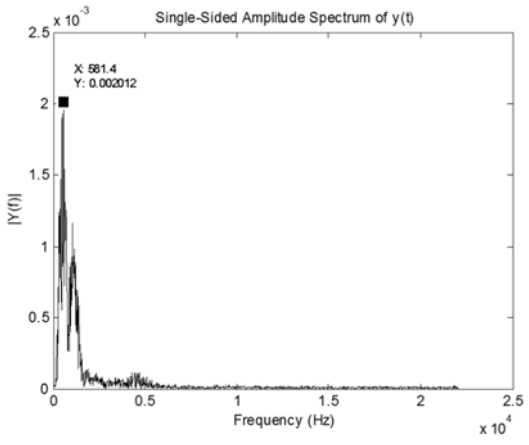
Soft food has different properties from crispy food. As chewing events increase, there is no significant change in the frequency domain because the sound doesn't change in the time domain. Figure 4.9 shows that the peak frequency moves around 500 Hz with 10~20% variation.



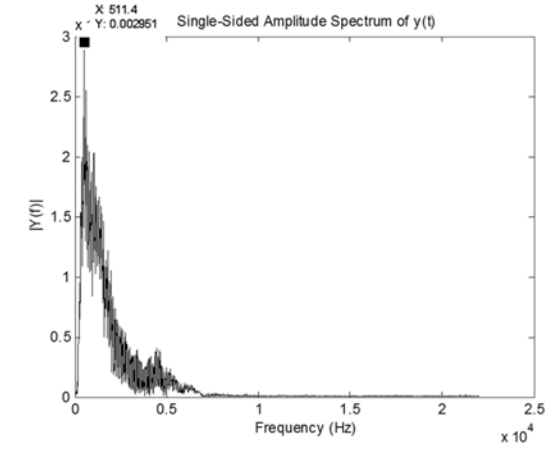
2nd bite, Pitch : 608 Hz



3rd bite, Pitch : 506 Hz



10th bite, Pitch : 581 Hz

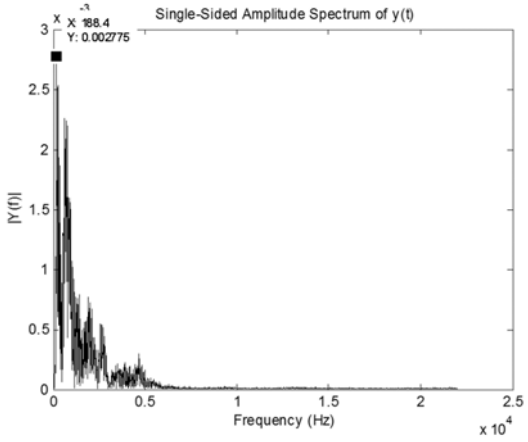


15th bite, Pitch : 511 Hz

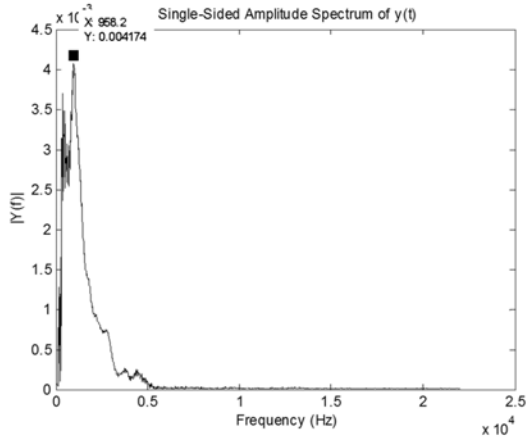
Figure 4.9. *Power spectrum of bread chewing sounds with arbitrary chewing*

In the two previous cases, the texture was the same for one piece or several pieces. In general, there were a lot more food kinds in addition to the previous two cases. If there are two more different ingredients such as pepperoni pizza and cheeseburger. When it is eaten, the piece that is eaten has one or more ingredients. Also, it contains crispy parts or soft parts. It has different sound patterns as different parts of the food is chewed.

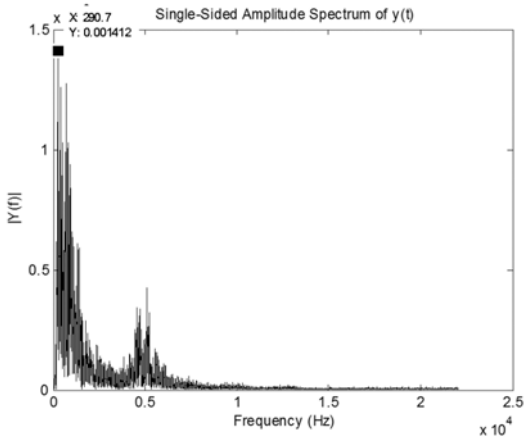
In this case, it has a random peak pattern because chewing parts are randomly selected. Figure 4.10 shows a random pattern.



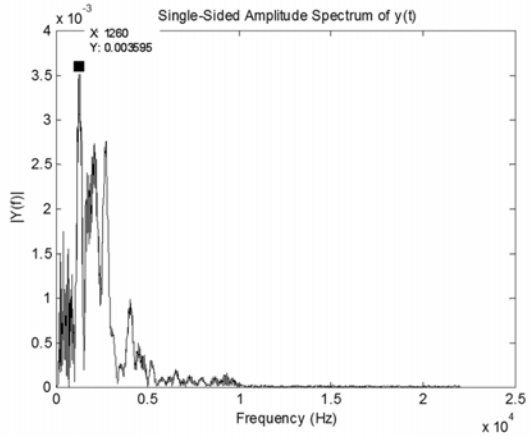
2nd bite, Pitch : 188 Hz



3rd bite, Pitch : 958 Hz



10th bite, Pitch : 290 Hz



15th bite, Pitch : 1260 Hz

Figure 4.10. *Power spectrum of cheeseburger chewing sounds with arbitrary chewing*

For this reason, algorithms are required to distinguish these different patterns. The first algorithm that I suggest is removing significantly different patterns in one swallowing cycle. The idea is to acquire all segments from the segmentation step and remove the very first segment and the last 20 or 30% of segments. If it is hard to decide to remove a segment, average the peak frequencies in the first few segments and set the variance threshold. This will be useful for crispy food but less effective for soft and other noncrispy food.

The second algorithm is voting for each chewing segment with the previous algorithm. For every segmented sound data, select the segmented sound with its majority. If a lower frequency segment is major then select the lower frequency segment. But if the ratio of high and low frequency segments is near 1, then select all high and low segment data. This algorithm is useful to classify using a decision tree. The next chapter shows a texture based decision tree with various classification algorithm.

4.2 Time based features

The explicit features in time domain are energy, pattern of energy variation with chewing events and variance of energy. Figures 4.11, 4.12, and 4.13 show enveloped signals for carrot (crispy), bread (soft) and pepperoni pizza (crispy and soft).

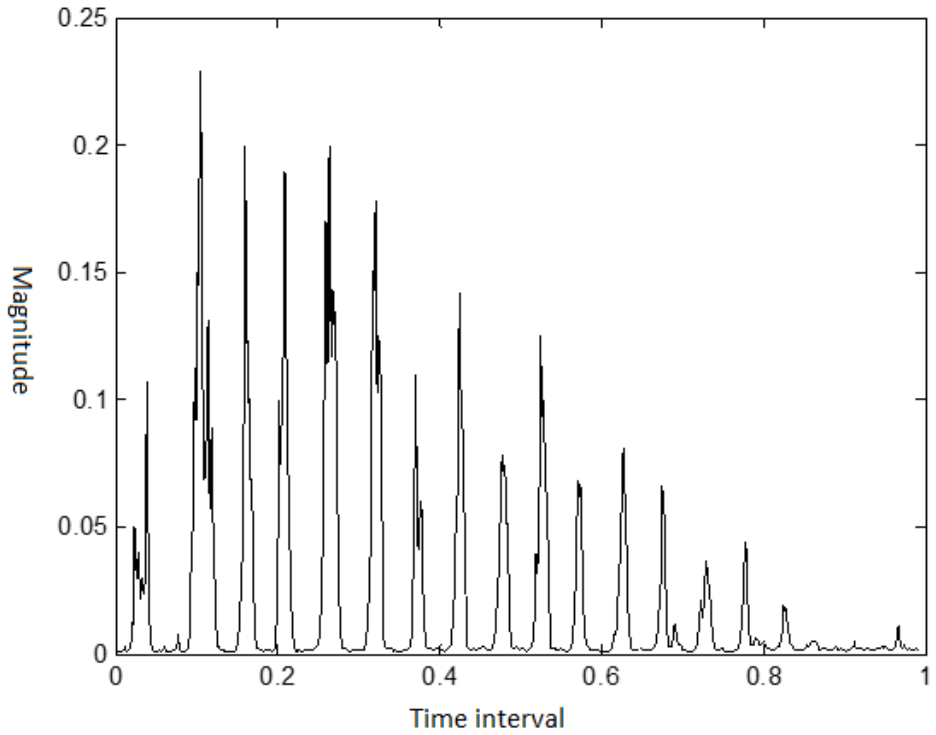


Figure 4.11. *Envelope of chewing signal of potato carrot*

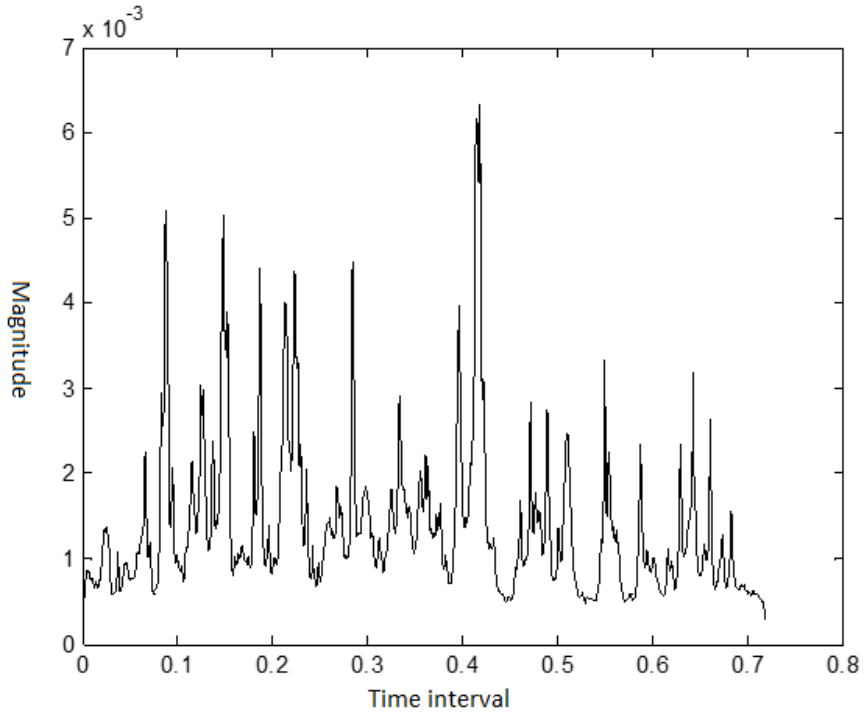


Figure 4.12. *Envelope of chewing signal of bread*

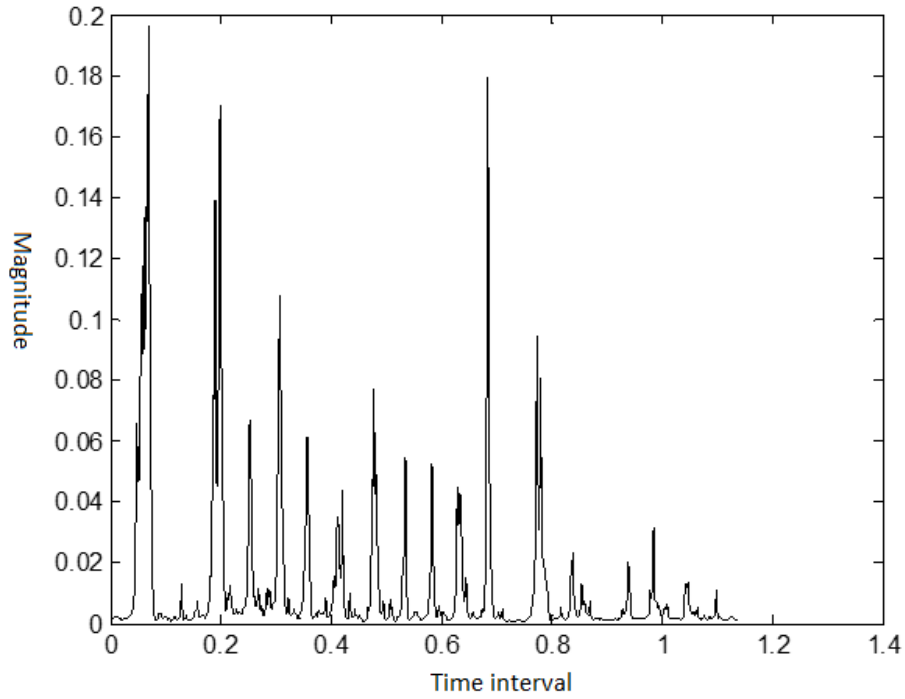


Figure 4.13. *Envelope of chewing signal of pepperoni pizza*

Carrot has a crispy texture. The energy per time unit is larger than others. It shows a monotonic decreasing pattern as chewing events are added. The variance is also larger than the others.

Bread has a soft texture. The energy per unit time is the smallest. The peaks show a random pattern. But the variance is small.

In the case of pepperoni pizza, the energy per unit time is smaller than for soft food and less than for crispy food. The pattern varies randomly and the variance is large.

	Energy	Energy pattern	Variance	Peak frequency	Frequency pattern
Crispy (Carrot)	High	Decrease (monotonic)	High	High	Decrease (monotonic)
Soft (Bread)	Low	Random	Low	Low	Constant (10–20%)
Mix (Pizza)	High	Random	High	High-Low	Random

Table 4.1 *Comparison with time domain data and frequency data for different chewing textures*

Bite sound energy

Bite sound energy represents energy of the bite segment signal. If it is larger, it can be a crispy food.

$$E_1 = \frac{\sum_{i \in Bite} |y_i|}{x_i}$$

Energy pattern

Energy pattern represents the peak variation pattern acquired from the signal envelope. Crispy food shows a decreasing pattern and the others show random or constant patterns. To calculate the pattern, I used the ratio between variances of increasing and decreasing.

$$E_2 = \frac{\sum_{i \in increase} |\Delta y_i|}{\sum_{j \in decrease} |\Delta y_i|}$$

If the ratio is close to zero, it represents a steep decreasing pattern. If it is close to one, it can be a random or constant pattern.

Variance

Variance is used with the bite energy and energy patterns. Table 4.1 shows crispy and mixed items show higher variance and soft items show lower variance.

$$E_3 = \mathrm{Var}(x) = \sum_{i=1}^{n} (x_i - \mu)^2$$

Chapter 5

Classification and identification

5.1 Background

To make right identification of food type is one of the goals of this research. Classification is the most important point of this paper. For this reason, three classification methods have been evaluated and their performance compared.

According to experimental data, there were 12 subjects, 10 solid foods and 3 or more chewing event cycles. So the total number of samples can be more than 360 but only 360 samples were picked and used for analysis. In 360 samples, 90 samples were recorded in noisy conditions and the rest of them were recorded in quiet conditions.

From the 270 samples which were recorded in a quiet place, 180 samples were used for training and rest of the 90 samples were used for identification.

Order	Food	Chewing texture
1	Bread	Soft
2	Burger	Soft
3	Carrot	Crispy
4	Cereal	Crispy
5	Chicken	Soft
6	Potato chips	Crispy
7	Cookie	Crispy
8	Potato fries	Soft
9	Lettuce	Crispy
10	Pizza	Soft

Table 5.1. *Selected foods and their chewing textures*

If people make the same chewing sound for the same food, one reference sample is good enough for classification. But, as shown in Fig. 5.1, the coefficients have a boundary. If the boundary is small enough to classify different samples, every participant doesn't need to have his/her own reference. But if the boundaries overlap, all participants must have their own reference sample. The third step is classification and recognition. Amft [8] used the Naïve Bayes method. It is a simple probabilistic classification. It uses an independent feature model. It is derived from Bayes' theorem. I used ANN (Artificial Neural Network) [22] and SVM (Support Vector Machine) [19] to find out the best algorithm to classify the sounds. All these algorithms are widely used for classification.

This paper uses two classification methods. One is K-Nearest Neighbor and the other is Naïve Bayes algorithm. For the training step, I used two subjects' data. And for the identification step, I used three other subjects' data.

Figs. 5.1 and 5.2 show the easiest and hardest data sets to draw classification boundaries. These samples show how easy or hard it is to classify the food from individual chewing sounds. The later part of this paper shows all accuracy for binary and multiple classifiers.

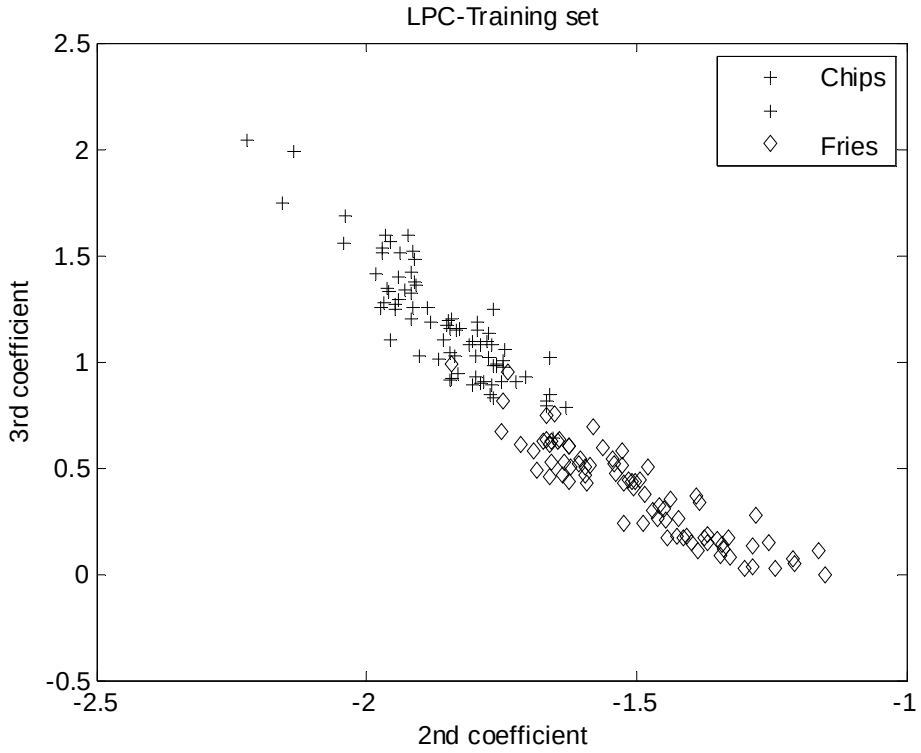


Figure 5.1 *Two groups for which it is easy to draw a classification boundary*

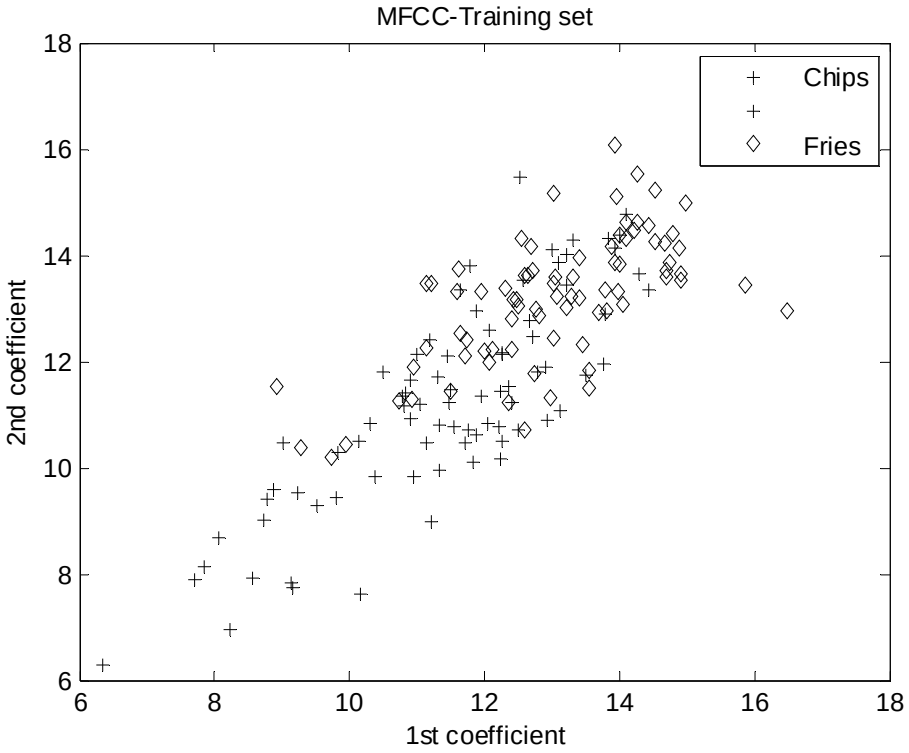


Figure 5.2. *Two groups for which it is hard to draw a classification boundary*

5.2 Classical classification/identification

K-Nearest Neighbors

K-Nearest Neighbors is a kind of nonparametric algorithm. KNN stores all the given data examples $\{fv_i,\ l_i\}$, where fv_i denotes the feature vectors in our pruning system and l_i denotes the class label. It uses these examples to estimate l_{new} for the new example fv_{new}. The l_{new} is assigned to the class having the largest representatives amongst the K nearest examples, which fv_{new} are similar to. Here we use NK to denote the number of nearest neighbors to distinguish the number of codewords in one codebook K.

The procedure of K-NN is as follows:

1. Instead of building a model, all the training examples $\{fv_i,\ l_i\}$ are stored.

2. Calculate the similarity between the new example fv_{new} and all the examples in the training set fv_i.

3. Determine the K-nearest examples to fv_{new}.

4. Assign l_{new} to the class that most of the K-nearest examples belong to.

Commonly the similarity between examples refers to the Euclidean distance, and the Euclidean distance between example vectors $M = (m1, m2,\ldots, mj)$ and $N = (n1, n2,\ldots, nj)$ is defined as:

$$d_\varepsilon(M,N) = \sqrt{\sum_{i=1}^{j}(m_i - n_i)^2}$$

where j is the dimension of vectors.

K-NN (Nearest Neighbors)

Table 5.2 and 5.3 show confusion matrix of KNN classification/identification. There are a couple of terms to define from the matrix.

- Accuracy $= \frac{TP+TN}{TP+FP+FN+TN}$
- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$

as, TP is True Positive, FP is False Positive, TN is True Negative and FN is False Negative.

Actual class	714	483	214	538	428	64	178	659	388	468	17.3	Bread
	760	681	289	487	506	148	349	745	289	476	14.4	Burger
	354	176	911	260	314	577	607	322	311	586	20.6	Carrot
	286	257	132	222	181	176	252	230	285	366	9.3	Cereal
	845	628	362	436	576	98	409	534	488	423	12.0	Chicken
	74	76	461	258	89	1330	812	209	243	283	34.7	Chips
	286	249	555	362	293	548	555	353	407	413	13.8	Cookie
	613	539	254	267	517	148	265	523	272	507	13.4	Fries
	356	355	272	354	346	359	488	315	455	438	12.2	Lettuce
	448	417	467	539	513	438	682	430	440	551	11.2	Pizza
	15.1	17.6	23.3	6.0	15.3	34.2	12.1	12.1	12.7	12.2	15.9	Precision
	Predicted class										Recall	

Table 5.2. *Identification rate of applying K-Nearest Neighbor for quiet environment recording. It shows confusion matrix of KNN ouput when the number of neighbor is 35.*

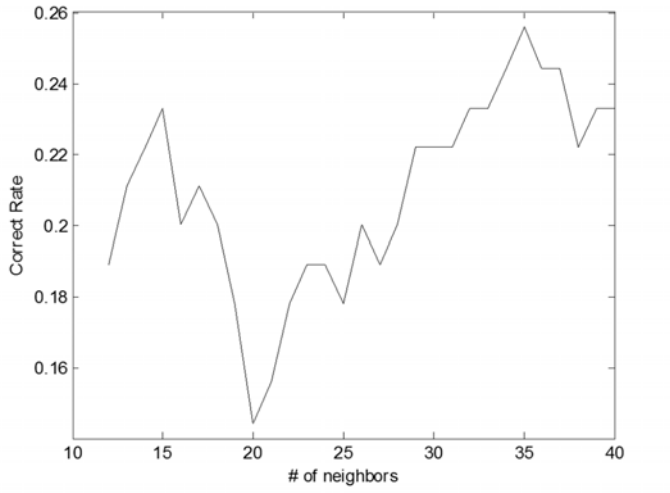


Figure 5.1 *Correct identification rate when # of neighbors are changed*

Actual class	245	166	51	121	137	27	36	106	136	76	22.3	Bread
	313	300	72	148	287	40	104	213	69	211	17.1	Burger
	111	162	308	87	148	220	187	154	151	189	17.9	Carrot
	99	88	88	45	54	53	90	48	63	76	6.4	Cereal
	325	257	153	191	190	17	143	302	121	258	9.7	Chicken
	34	37	137	139	33	300	173	50	72	149	26.7	Chips
	75	92	107	150	85	192	167	64	247	105	13	Cookie
	113	139	47	73	185	72	52	170	90	100	16.3	Fries
	114	71	160	90	129	67	90	112	799	94	46.3	Lettuce
	183	186	119	163	231	99	172	187	217	163	9.5	Pizza
	15.2	20	24.8	3.7	12.8	27.6	13.8	12.1	40.7	11.5	19	Precision
	Predicted class										Recall	

Table 5.3. *Identification rate of applying K-Nearest Neighbor for noisy environment recording. It shows confusion matrix of KNN ouput when the number of neighbor is 12.*

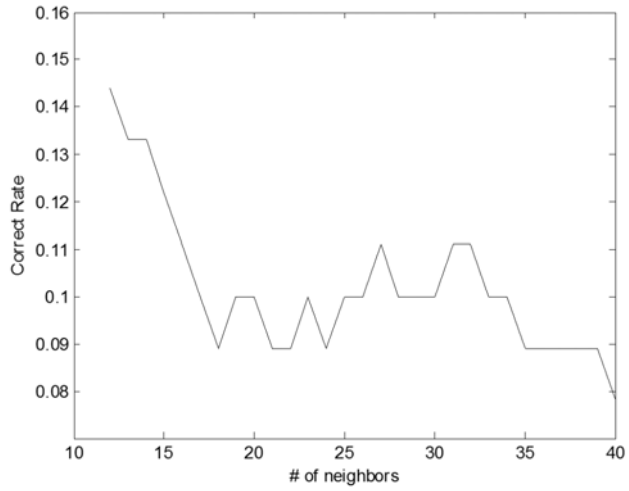


Figure 5.3 *Correct identification rate when # of neighbors are changed*

Naïve Bayes

From Bayes' theorem

$$\mathrm{p}(C|F_1, \dots F_n) = \frac{\mathrm{p}(C)\mathrm{p}(F_1, \dots, F_n|C)}{\mathrm{p}(F_1, \dots, F_n)}$$

The conditional probability has been estimated from the previous equation. It can be calculated from discrete data or the continuous distribution function.

For example, suppose the training data contain a continuous attribute, *x*. We first segment the data by the class, and then compute the mean and variance *x* of in each class. Let μ_c be the mean of the values in associated with class *c*, and let σ_c^2 be the variance of the values in *x* associated with class *c*. Then, the probability of some value given a class, $P = (x = v|c)$, can be computed by plugging *v* into the equation for a Normal distribution parameterized by μ_c and σ_c^2. That is,

$$\mathrm{P} = (x = \mathrm{v}|\mathrm{c}) = \frac{1}{\sqrt{2\pi\sigma_{\mathrm{c}}^2}}\mathrm{e}^{-\frac{(\mathrm{v}-\mu_{\mathrm{c}})^2}{2\sigma_{\mathrm{c}}^2}}$$

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule. The corresponding classifier is the function defined as follows

$$\mathrm{classify}(\mathrm{f}_1, \ldots \mathrm{f}_{\mathrm{n}}) = \mathrm{argmax}\, \mathrm{p}(C = \mathrm{c}) \prod_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{p}(\mathrm{F}_{\mathrm{i}} = \mathrm{f}_{\mathrm{i}}|C = \mathrm{c})$$

In this paper, the mean and variance of the training set have been acquired. Also, the acquired data are independent. So, the probability for each coefficient is multiplied.

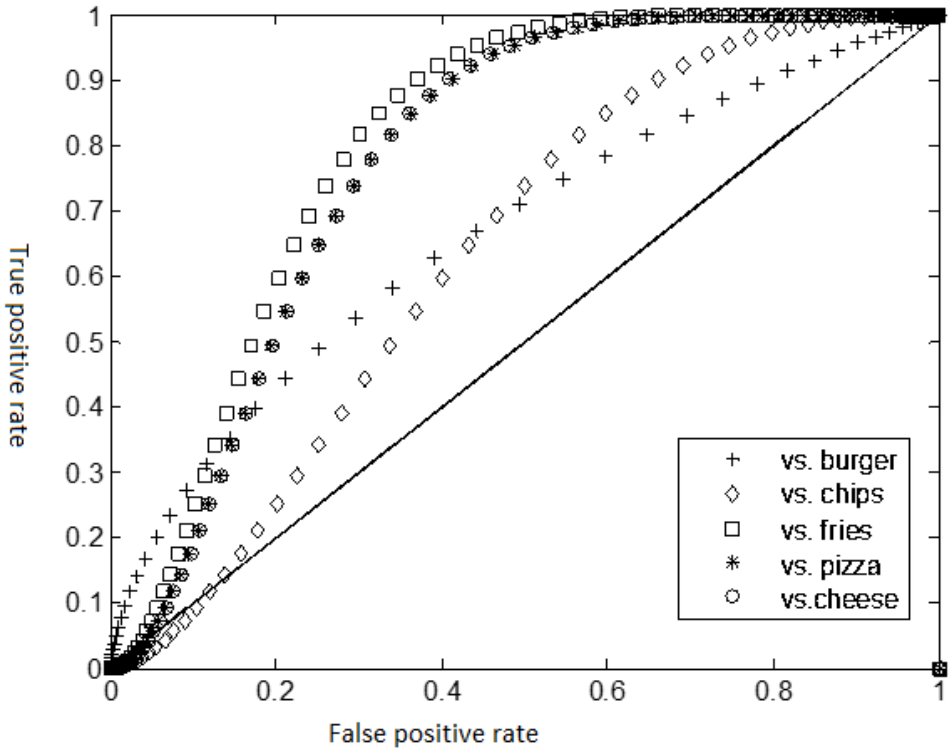


Figure 5.4. *ROC curves for apple vs. other foods (MFCC data)*

Fig. 5.4 shows the ROC curve for the binary classifier. There are 16 dimensions. For the 1st coefficient, the Gaussian distribution is used but the 2nd to 16th coefficients are randomly selected.

Neural Network

The first calculation model of the neural network is known from McCulloch's paper in 1943. Later, Hebb suggested the first learning algorithm in 1949. Rosenblatt [34] published the perceptron for the first time. But the perceptron has a limitation because it is a linear classifier and impossible to classify the XOR problem [35]. In 1986, Rumelhart [36] showed

a multiperceptron with a hidden layer and back propagation algorithm. This paper showed that a neural network can solve complex problems.

Figure 5.5 shows the structure of a multilayer perceptron. The left layer is the input layer and the right layer is the output layer. the input layer has n + 1 nodes. It is a vector that contains $x = (x_1, x_2, \cdots, x_n)$. The last node is a bias which always has the value 1. The output vector has k nodes that represent the Actual class. The hidden layer(s) is (are) located in between the input and output layer and has p + 1 nodes. The bias node is always 1.

There are (n + 1)*p nodes have variable weights between the input and hidden layer(s). It is written as vector V. Also, there are (p + 1)*k nodes shown as vector W.

The method to compute these two vectors are the main issue to train the neural network model. For the node vectors, an activation function is required. It can excite and suppress the sum of connection weights to activate nodes. For the first time, step functions have been used but a sigmoid function is used currently.

Figure 5.6 shows a concept sigmoid function for perceptron and activation nodes.

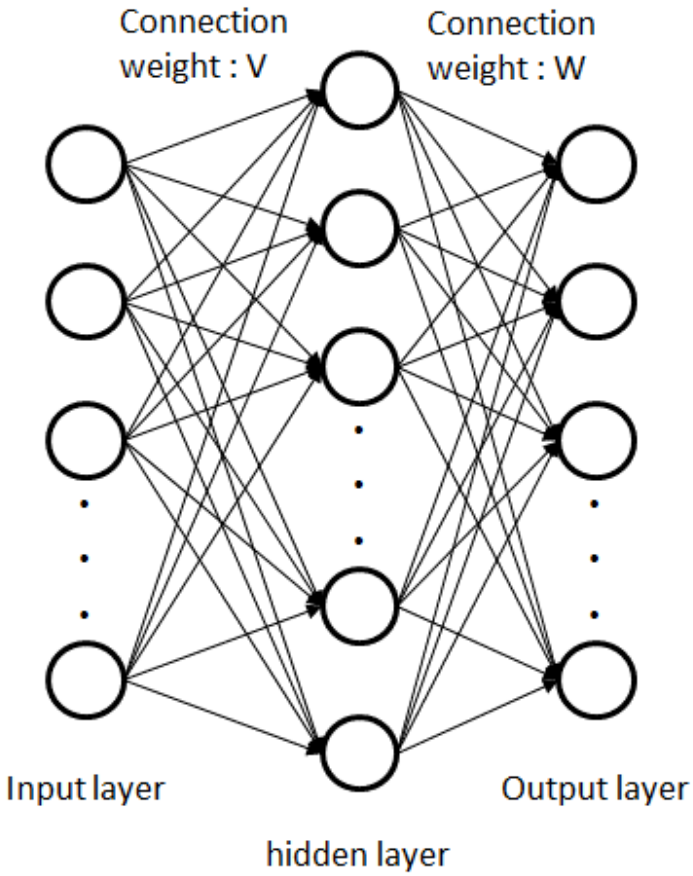


Figure 5.5. *Structure of neural network with multilayer perceptron.*

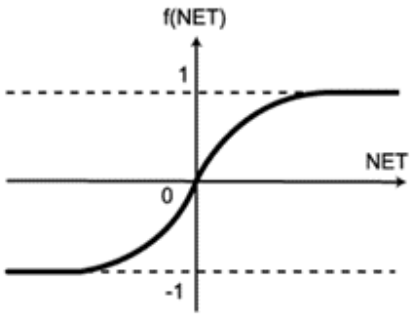


Figure 5.6. *Weight function for hidden and output layers.*

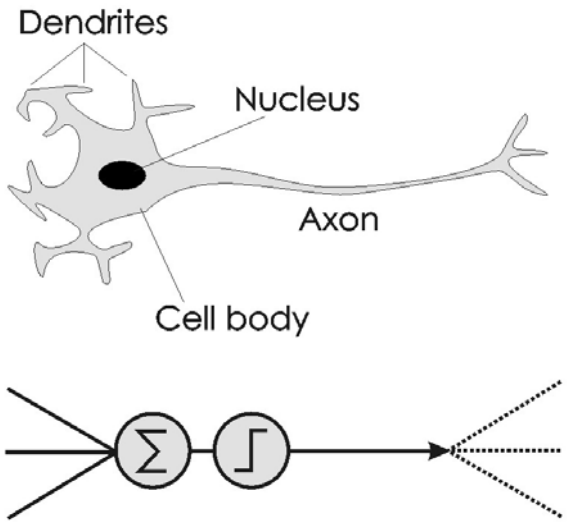


Figure 5.7 *Structure of neuron and neural network for a single neuron with weight function*

We compute these node weights as a training step. The back propagation algorithm is used for computing the weights.

The first step is initializing node weights using random numbers and loop count. The random variables are within –0.5 to 0.5. Before computing the weight, we must decide the features for the input layer, number of hidden layers and elements of the output layer. The input layer is decided by the number of features and the output layer is decided by the Actual class. For example, if there are 10 feature vectors and 5 Actual classes, the model has 10 elements of input layer and 5 elements of output layer. The number of hidden layers can be selected as an arbitrary number but if it is too small, it is hard to make complex classifications and if it is too large, it takes a long time to compute them.

When all variables are initialized, the next step is to compute the output of the hidden layer and output layer. Two equations are used for the calculation.

$$\mathrm{NET_z} = \mathrm{XV^T}\ ,\ \mathrm{z} = \mathrm{f(NET_z)} = \frac{1}{1 + e^{\mathrm{NET_z}}}$$

$$\mathrm{NET_y} = \mathrm{ZW^T}\ ,\ \mathrm{y} = \frac{1}{1 + e^{\mathrm{NET_y}}}$$

For these equations, the bipolar sigmoid function has been applied as an activation function. Each iteration computes the error and it adjusts weight vectors.

For the final condition, I suggest cross validation using validation sets. As iteration proceeds, the error will decrease. It also decreases error for validation set. The minimal point of validation error is the stop point for iterations.

Table 5.4 shows the process.

Step 1. Initialize weights and counter : V, W = random numbers (–0.5~0.5)

Step 2. Set learning rate and error

Step 3. Compute output of hidden layer and output layer

$\mathrm{NET_z} = \mathrm{XV^T}$, $\mathrm{z} = \mathrm{f(NET_z)} = \frac{1}{1+\mathrm{e^{NET_z}}}$

$\mathrm{NET_y} = \mathrm{ZW^T}$, $\mathrm{y} = \frac{1}{1+\mathrm{e^{NET_y}}}$

Step 4. Compute output error

$\frac{1}{2}(\mathrm{d}-\mathrm{y})^2 + \mathrm{E}$

Step 5. Compute error signal of output error

$\delta_\mathrm{v} = (\mathrm{d}-\mathrm{v})\mathrm{v}(1-\mathrm{v})$

Step 6. Compute error signal of hidden layer

$$\delta_\mathrm{y} = y(1-y)\sum_{\mathrm{i}=1}^{\mathrm{m}} \delta_\mathrm{v}\mathrm{w}$$

Update weight

$$\mathrm{W} = \mathrm{W} + \alpha\delta_\mathrm{v}\mathrm{Z}$$

$$\mathrm{V} = \mathrm{V} + \alpha\delta_\mathrm{y}\mathrm{X}$$

Step 7. Increase counter

Step 8. Test stop condition (less than predetermined Error)

If larger error, go to step 3

Step 9. Identification for validation data set

Step 10. Determine local minimum of validation error

Table 5.4 *Compute procedure of neural network*

In this research, we used 19 elements of hidden layer, 17 elements of input layer and 10 elements of output layers. 135 samples were used for training and 45 samples were used for validation. For testing, 90 samples were used. Table 5.5 shows the summary of parameters.

Number of	
Hidden layers	19 (found max)
Input parameters	17
Output (Classes)	10
Training samples	135
Validation samples	45
Testing samples	90

Table 5.5. *Parameters for evaluation of neural network*

The stop condition of training is the minimum point of the validation error. In the first analysis for quiet condition samples shows the error pattern in figure 5.8. Tables 5.6 and 5.7 show identification results using a Neural Network with quiet and noisy environment recordings. The result doesn't have specific characteristics. Figures 5.9 and 5.10 show their ROC curves.

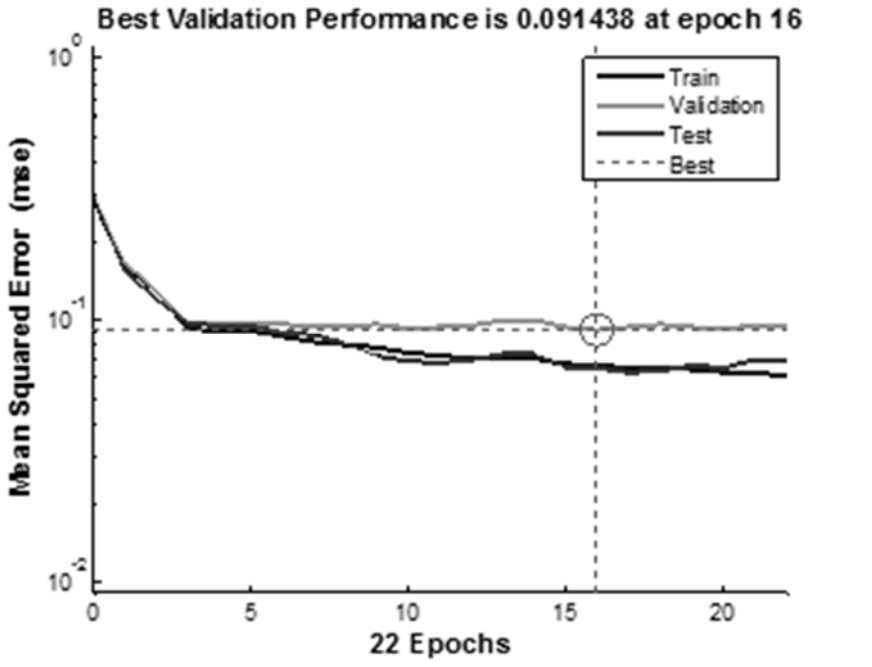


Figure 5.8. *Stop condition of neural network training with quiet condition experimental data*

Actual class												
	603	1039	221	136	1021	10	109	751	122	122	14.6	Bread
	941	908	236	127	1268	0	145	640	261	204	19.2	Burger
	246	565	1463	74	258	588	294	268	257	405	33.1	Carrot
	466	355	267	62	249	52	165	402	178	191	2.6	Cereal
	937	1038	276	175	723	36	116	596	504	399	15.1	Chicken
	87	114	1104	12	169	1357	268	214	271	239	35.4	Chips
	247	483	791	22	299	431	633	512	333	270	15.7	Cookie
	684	999	247	111	617	44	163	678	183	179	17.4	Fries
	382	475	516	48	411	239	199	491	729	248	19.5	Lettuce
	533	617	835	118	1054	202	299	307	484	476	9.7	Pizza
	11.8	13.8	24.6	7.0	11.9	45.9	26.5	14.0	21.9	17.4	18.7	Precision
	Predicted class										Recall	

Table 5.6. *Confusion matrix of neural network identification rate with quiet environment recording data*

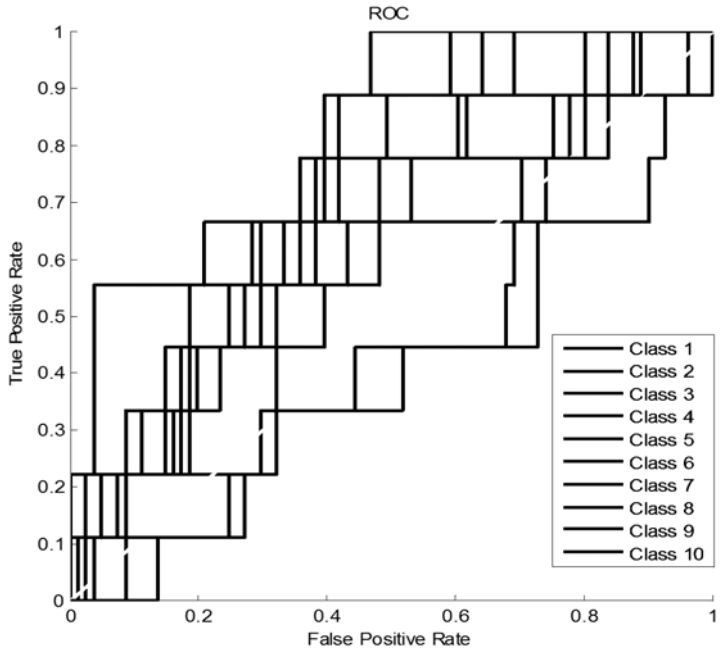


Figure 5.9. *ROC of neural network identification rate with quiet environment recording data*

Actual class	444	349	82	0	0	0	0	62	60	104	40.3	Bread
	582	696	0	0	0	33	0	50	177	219	39.6	Burger
	189	230	607	16	4	177	0	5	220	269	35.4	Carrot
	67	204	129	0	0	43	0	21	77	163	0	Cereal
	810	656	160	25	0	19	0	40	72	176	0	Chicken
	18	76	146	1	0	583	0	1	156	143	51.9	Chips
	62	220	242	0	0	265	0	2	244	112	0	Cookie
	300	407	73	0	0	37	0	68	44	112	65	Fries
	305	251	205	0	0	147	0	32	456	330	26.4	Lettuce
	237	403	204	36	0	63	0	136	281	360	20.9	Pizza
	14.7	19.9	32.8	0	0	42.6	0	16.3	25.5	16.9	22.7	Precision
	Predicted class										Recall	

Table 5.7. *Confusion matrix of neural network identification rate with noisy environment recording data*

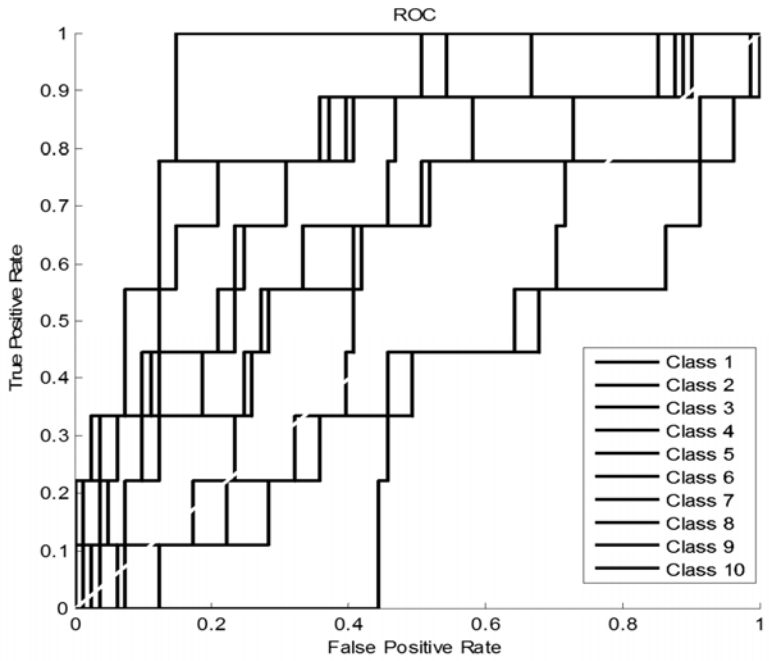


Figure 5.10. *ROC of neural network identification rate with noisy environment recording data*

Support Vector Machine

SVM (Support Vector Machine) was proposed by Vladimir Vapnik in late 1992. The main idea of this algorithm is finding a decision boundary which is far from the points in the training data.

The goal of other classification algorithms is minimizing the error rate. But the goal of the SVM is maximizing the margin among training groups. The SVM is based on the linear discriminant function.

The classifier sets the feature vector to w_i if it satisfies the condition,

$$g_i(x) > g_j(x) \quad \text{for all } i \neq j$$

If there are two groups,

$$g(x) \equiv g_1(x) - g_2(x)$$

Decide using w_1 if $(x) > 0$; otherwise decide using w_2

$g(x)$ is the linear discriminant function and is defined by

$$g(x) = w^T x + b$$

It represents a hyperplane that divides feature space. Figure 5.11 shows the hyperplane and margin in two dimensional space.

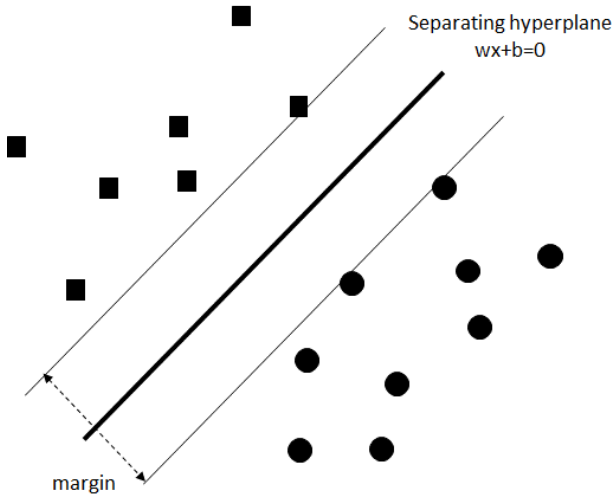


Figure 5.11. *Hyperplane and its margin for two different classes*

The distance between arbitrary points to the hyperplane is

$$\mathrm{n} = \frac{\mathrm{w}}{\|\mathrm{w}\|}$$

It can be classified into two groups.

$\mathrm{w}^{\mathrm{T}}x + \mathrm{b} > 0$: Group 1

$\mathrm{w}^{\mathrm{T}}x + \mathrm{b} < 0$: Group 2

The problem is how to separate the two groups using a linear discriminant function which has a minimum error rate. There are infinite number of answers. But there is only one answer which has the maximum margin. The definition of margin is the thickness between two closed data points for each group.

$$\text{For } y_i = +1, \ \mathrm{w}^{\mathrm{T}}x_i + \mathrm{b} > 0$$

$$\text{For } y_i = -1, \ \mathrm{w}^{\mathrm{T}}x_i + \mathrm{b} < 0$$

They can be rewritten to change the scale,

$$\text{For } y_i = +1, \ \mathrm{w}^{\mathrm{T}}x_i + \mathrm{b} \geq 1$$

$$\text{For } y_i = -1, \ \mathrm{w}^{\mathrm{T}}x_i + \mathrm{b} \leq -1$$

There are two assumptions,

$$\mathrm{w}^{\mathrm{T}}x^{+} + \mathrm{b} = 1$$

$$\mathrm{w}^{\mathrm{T}}\mathrm{x}^{-} + \mathrm{b} = -1$$

The width of the margin is calculated by

$$\mathrm{M} = (x^{+} - x^{-}) \times \mathrm{n} = (x^{+} - x^{-}) \times \frac{\mathrm{w}}{\|\mathrm{w}\|} = \frac{2}{\|\mathrm{w}\|}$$

The problem is how to maximize the margin (distance) between the two groups. So

Maximize $\frac{2}{\|\mathrm{w}\|}$ is minimize $\frac{1}{2}\|\mathrm{w}\|^2$ at the condition,

$$y_i(\mathrm{w}^{\mathrm{T}}x_i + \mathrm{b}) \geq 1$$

To solve this problem, apply the Lagrange multiplier.

$$\text{minimize } \mathrm{L_p}(\mathrm{w}, \mathrm{b}, \alpha_{\mathrm{i}}) = \frac{1}{2}\|\mathrm{w}\|^2 - \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}}(y_i(\mathrm{w}^{\mathrm{T}}x_i + \mathrm{b}) - 1)\,, \alpha_{\mathrm{i}} \geq 0$$

$$\frac{\partial \mathrm{L_p}}{\partial \mathrm{w}} = 0\,, \mathrm{w} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}} y_i x_i$$

$$\frac{\partial \mathrm{L_p}}{\partial \mathrm{b}} = 0\,, \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}} y_i = 0$$

$$\text{maxmize } \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}} - \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{n}}\sum_{\mathrm{j}=1}^{\mathrm{n}} \alpha_{\mathrm{i}}\alpha_{\mathrm{j}} y_i y_j x_i^T x_j\,, \alpha_{\mathrm{i}} \geq 0, \text{and } \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}}\, \mathrm{y_i} = 0$$

From this equation, we apply the Karush-Kuhn-Tucker (KKT) condition.

$$\alpha_{\mathrm{i}}(y_i(\mathrm{w}^{\mathrm{T}}x_i + \mathrm{b}) - 1) = 0$$

Only the support vector has this condition.

$$\alpha_{\mathrm{i}} \neq 0$$

The solution has the form,

$$\mathrm{w} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}} y_i x_i = \sum_{\mathrm{i} \in \mathrm{SV}} \alpha_{\mathrm{i}} y_i x_i$$

So, the linear discriminant function is defined by

$$\mathrm{g}(x) = \mathrm{w}^{\mathrm{T}} x + \mathrm{b} = \sum_{\mathrm{i} \in \mathrm{SV}} \alpha_{\mathrm{i}} x_i^T + \mathrm{b}$$

This requires a solution of a linear separable problem. If it is not separable, a slack variable can be added to allow classification. A slack variable is defined by the distance between incorrectly classified data and the boundary of the class.

$$y_i(\mathrm{w}^{\mathrm{T}} x_i + \mathrm{w}_0) \geq 1 - \xi_{\mathrm{i}}$$

Figure 5.12 represents the slack variable and how it can be seen in two dimensional space.

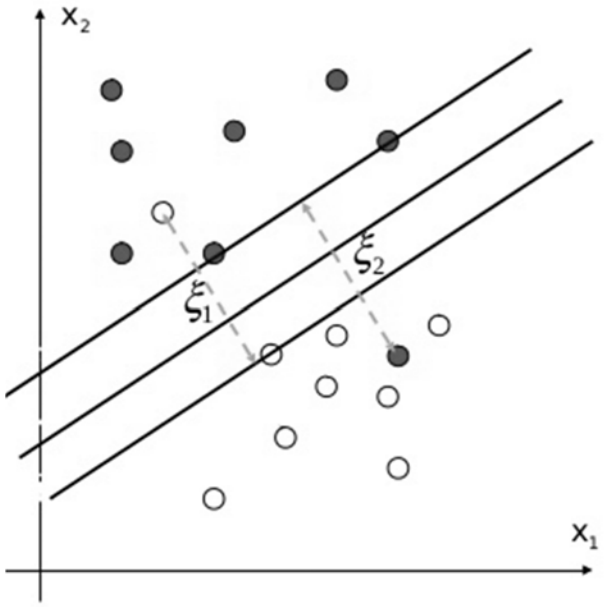


Figure 5.12. *Binary classified data with slack variables. The two groups are not perfectly classified because it is impossible to split using the linear discriminant function. For unclassified data, we apply slack variables.*

To apply the Lagrange multiplier, it has to minimize the equation,

$$\frac{1}{2}\|\mathrm{w}\|^2 + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{n}} \xi_{\mathrm{i}}$$

and satisfy this condition

$$y_i(\mathrm{w}^{\mathrm{T}} x_i + \mathrm{b}) \geq 1 - \xi_{\mathrm{i}}\,, \xi_{\mathrm{i}} \geq 0$$

So, it has to solve this problem which is the same as the previous problem that doesn't have the slack variable.

$$\sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}} - \frac{1}{2}\sum_{\mathrm{i}=1}^{\mathrm{n}}\sum_{\mathrm{j}=1}^{\mathrm{n}} \alpha_{\mathrm{i}}\alpha_{\mathrm{j}} y_i y_j x_i^T x_j\,, \text{ and } \sum_{\mathrm{i}=1}^{\mathrm{n}} \alpha_{\mathrm{i}}\, y_i = 0$$

The difference is the condition of α_i. It has to be satisfied by

$$0 \leq \alpha_{\mathrm{i}} \leq \mathrm{C}$$

Kernel function

If the problem is too hard to solve by applying a linear function, it can map to a higher dimension. The Kernel function is a higher dimension mapping function. Calculation is performed in lower dimension but the classification is performed in higher dimension.

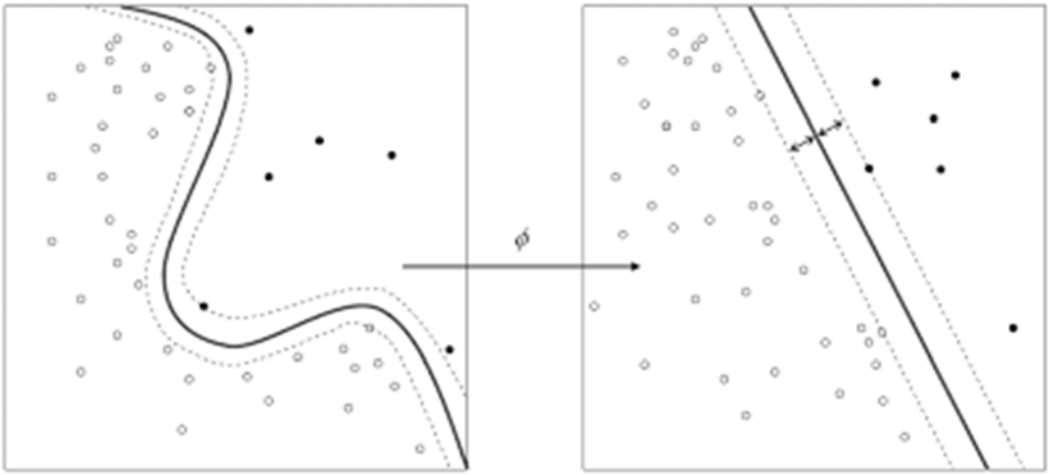


Figure 5.13. *Classified data by discriminant function with kernel function. Although the data can't be classified by LDF (Linear discriminant function), it can be classified using the kernel function. The kernel function maps lower dimension data to higher dimension data but the calculation is simple because it uses a dot product.*

Figure 5.13 represents how the kernel function works. The function classifies two groups in higher dimensions but it performs calculations in a low dimension.

It can be defined by

$$g(x) = w^{T}\phi(x) + b = \sum_{i \in SV} \alpha\, \phi(x_i)^{T}\phi(x) + b$$

The Kernel function is defined by a dot product with two feature vectors in a certain feature space.

$$K(x_i, x_j) \equiv \phi(x_i)^{T}\phi(x_j)$$

It can also apply the Lagrange multiplier.

$$\text{maxmize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i K(x_i, x_j)$$

$$0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

The solution of the function is

$$g(x) = \sum_{i \in SV} \alpha_i K(x_i, x) + b$$

Table 5.8 shows four kernel functions that have been used before. If the function can be explained using a dot product, it can be a kernel function. The Gaussian kernel is used for classification of the experimental data.

Linear kernel	$\mathbf{k}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y})$
Polynomial kernel	$k(x, y) = (x \cdot y + c)^d$
Sigmoid kernel	$k(x, y) = \tanh(\theta_1 x \cdot y + \theta_2)$
Gaussian kernel	$k(x, y) = \exp\left\{-\frac{\|x - y\|^2}{2\sigma^2}\right\}$

Table 5.8. *Kernel functions*

Multi-SVM

1:(M – 1) classifier

The Multi-SVM uses an M binary classifier. The jth binary classifier separates w_i and the rest of the M – 1 group. To train this classifier, w_i is positive and the other M – 1 group is the negative training group. From the identification, there may be one positive value but

there is no guarantee. So it takes the maximum value. To write a simple equation, the M class SVM classifier with x is w_k

$$k = \text{argmax}\, d(x)$$

Table 5.9 and 10 show the confusion matrix using the SVM method.

Actual class	541	1001	163	90	715	26	164	856	332	246	13.1	Bread
	693	1486	234	161	678	24	383	619	272	180	31.4	Burger
	228	377	1005	192	241	712	491	364	261	547	22.7	Carrot
	463	380	236	104	278	73	213	241	165	234	4.4	Cereal
	782	934	286	190	787	130	99	884	388	319	16.4	Chicken
	94	176	861	93	138	1308	449	78	378	260	34.1	Chips
	397	322	434	194	333	475	819	310	352	385	20.4	Cookie
	759	711	190	114	572	70	419	571	195	304	14.6	Fries
	479	519	243	70	425	303	393	365	447	494	12.0	Lettuce
	616	742	499	291	384	376	509	421	461	626	12.7	Pizza
	10.7	22.4	24.2	6.9	17.3	37.4	20.8	12.1	13.7	17.4	18.8	Total
	Predicted class											

Table 5.9. *Confusion matrix of identification rate of SVM classifier with quiet environment recording data*

Actual class	291	206	110	40	153	18	48	107	95	33	26.4	Bread
	450	479	31	101	366	34	44	151	66	35	27.3	Burger
	140	158	434	6	99	227	206	174	189	84	25.3	Carrot
	88	57	106	22	90	28	40	106	76	91	3.1	Cereal
	617	306	101	77	287	19	84	320	64	82	14.7	Chicken
	4	29	217	1	92	465	72	32	183	29	41.4	Chips
	56	210	130	26	20	235	109	193	227	78	8.5	Cookie
	270	173	79	8	138	10	90	181	70	22	17.4	Fries
	192	60	200	34	131	50	133	302	443	181	25.7	Lettuce
	149	269	178	56	110	38	64	416	274	166	9.7	Pizza
	12.9	24.6	27.4	5.9	19.3	41.4	12.2	9.1	26.3	20.7	20.4	Total
	Predicted class											

Table 5.10. *Confusion matrix of identification rate of SVM classifier with noisy environment recording data*

5.3 Hybrid hierarchy structure (decision tree)

The previous classification/identification shows a very low identification rate. In this chapter, I suggest a hybrid hierarchy structure such as decision tree based classification with various algorithms.

There are some previous researches that use this hierarchy structure in the machine learning area. This hybrid method has been applied with classification algorithms such as Neural Network, Support Vector Machine and others. The previous research shows performance improvement. The structure is defined a by binary tree. Each node has two branches until all data are classified. Each node has its classification algorithm and the algorithm can be any type of binary classifier. Figure 5.14 represents a hybrid hierarchy classifier using the SVM method.

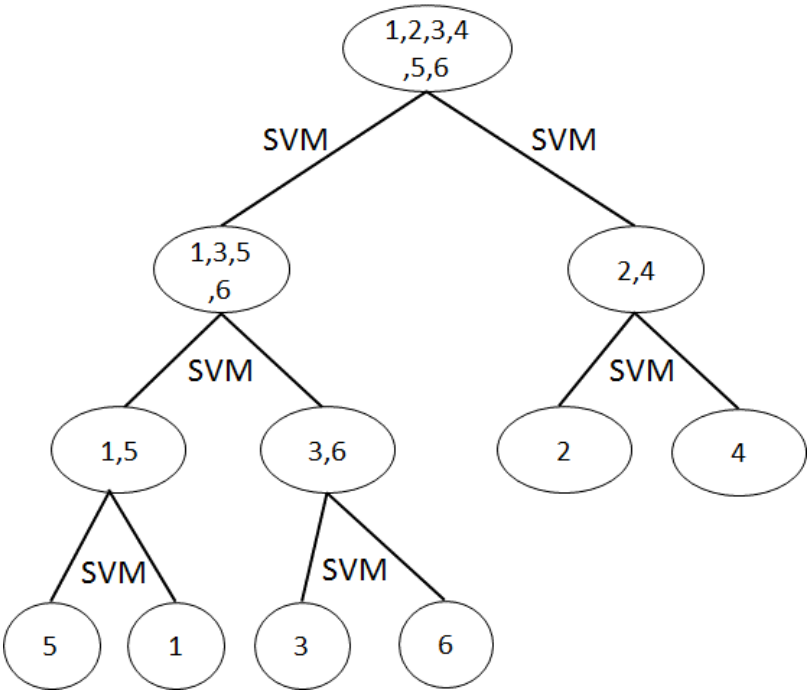


Figure 5.14. *General structure of hybrid hierarchy classifier with decision tree*

Parameters for the decision tree are food textures. In general, the most distinguishable and easiest parameters to identify from the chewing sounds are crispness and hardness. Less effective texture parameters are size and shape. For this reason, the textures such as crispness and hardness are selected to apply the decision tree. Figure 5.15 is the ideal design of a hybrid classifier. The mixed food can be placed at the next crispy node.

In the previous chapter, I already mentioned the feature with peak frequency and its randomness. In this chapter, I added three features in the time domain and applied them to the decision tree to increase the identification rate.

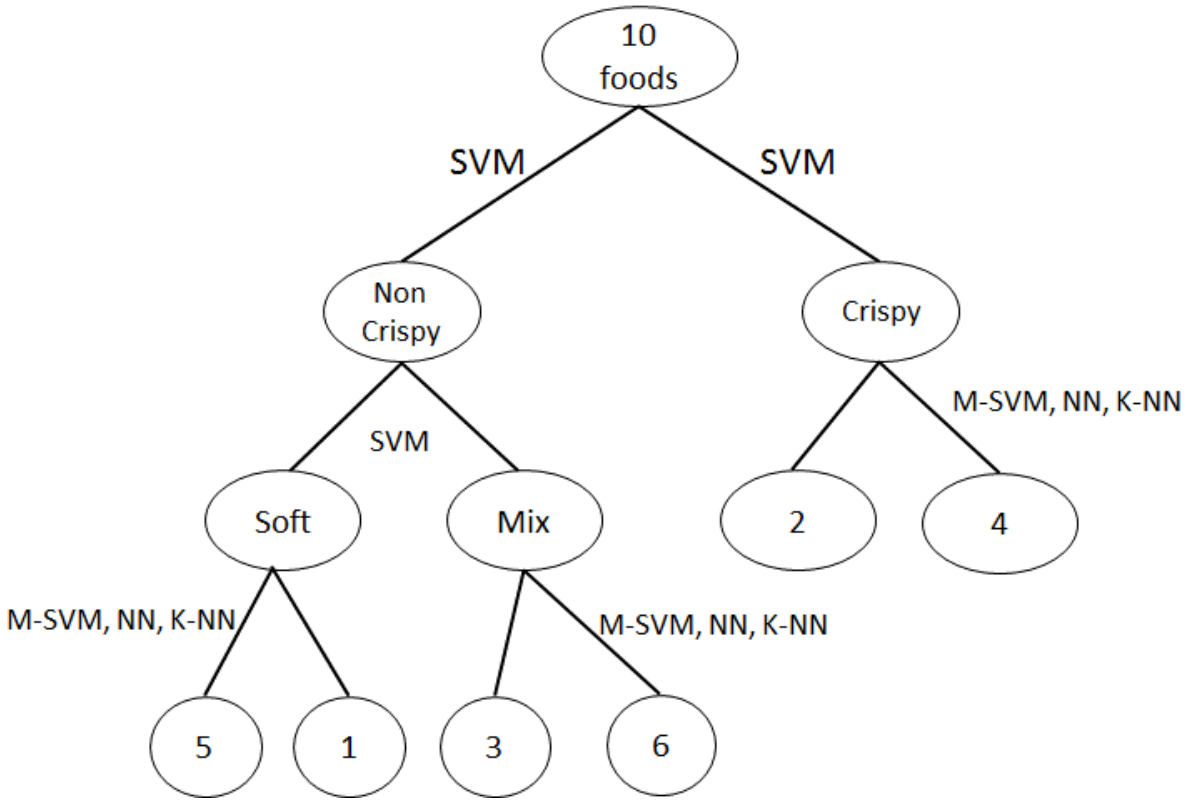


Figure 5.15. *Optimized structure of the hybrid hierarchy classifier for chewing sounds*

For the hybrid decision tree, these three variables are applied to decide crispy and noncrispy. Figure 5.16 is the structure of the hybrid classifier for this research. It has a simple structure. The tree doesn't select mixed food.

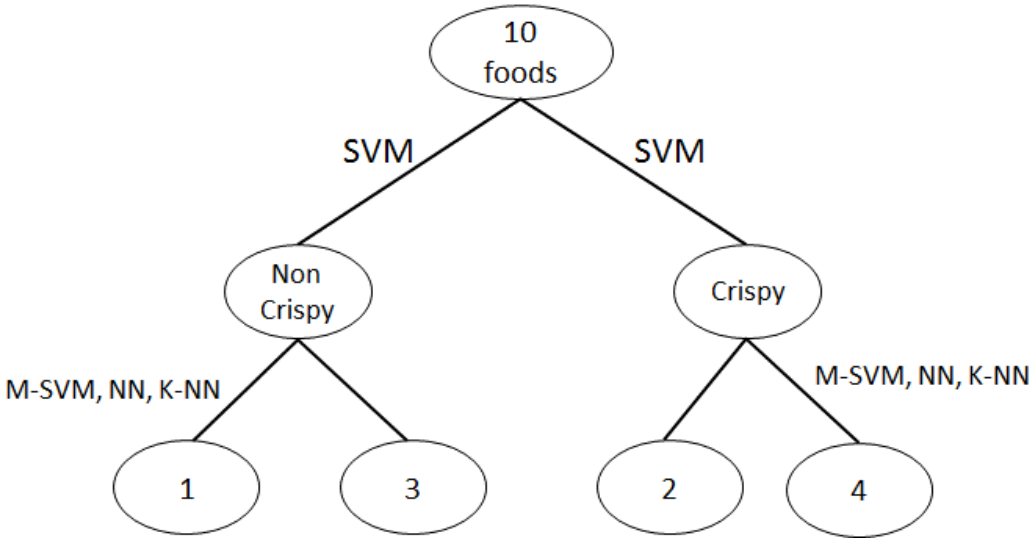


Figure 5.16. *The hybrid hierarchy classifier for the research*

The first part is level-one decision. It classifies 10 foods into two categories such as crispy and noncrispy food. K-NN is evaluated first.

The second trial algorithm is a neural network. It uses 10 hidden layers and it decides the lowest error rate in the validation set.

The third algorithm is SVM. It uses the RBF (Radial Basis Function) kernel. Figure 5.17 shows the result with the decision surface and support vectors. Table 5.11 shows confusion matrices which are applied by three classification methods with additional time domain features.

Actual class	6	1	0	2	0	0	0	0	0	0	0.67	Bread
	0	1	0	4	4	0	0	0	0	0	0.11	Burger
	0	0	9	0	0	0	0	0	0	0	1.00	Carrot
	1	1	0	2	1	0	1	2	0	1	0.22	Cereal
	1	3	0	1	1	0	1	1	0	1	0.11	Chicken
	0	0	6	0	0	1	1	0	1	0	0.11	Chips
	1	0	3	1	0	1	2	0	0	1	0.22	Cookie
	1	0	0	4	2	0	0	1	1	0	0.11	Lettuce
	0	0	0	4	4	0	0	1	0	0	0.00	Fries
	0	1	0	1	5	0	1	0	1	0	0.00	Pizza
	0.60	0.14	0.50	0.11	0.06	0.50	0.33	0.20	0.00	0.00	0.26	
	Predicted class											

(a) KNN output

Actual class	6	4	0	2	0	0	0	3	0	0	0.4	Bread
	3	0	0	0	3	0	1	1	1	1	0	Burger
	0	0	0	0	0	0	0	0	0	1	0	Carrot
	0	1	0	4	1	0	0	2	2	2	0.33	Cereal
	0	0	0	0	0	0	0	0	0	0	0	Chicken
	0	0	6	3	0	7	5	1	2	0	0.29	Chips
	0	0	0	0	0	0	0	0	0	0	0	Cookie
	0	2	0	0	0	0	0	2	0	2	0.33	Lettuce
	0	2	3	0	5	2	3	0	4	3	0.18	Fries
	0	0	0	0	0	0	0	0	0	0	0	Pizza
	0.67	0	0	0.44	0	0.78	0	0.22	0.44	0	0.26	
	Predicted class											

(b) Neural Network output

Actual class	2	4	0	2	0	0	0	0	0	1	0.22	Bread
	0	2	0	3	2	0	0	1	0	1	0.22	Burger
	0	0	1	1	0	2	1	0	2	2	0.11	Carrot
	1	0	0	5	0	2	0	0	0	1	0.56	Cereal
	0	0	0	1	5	0	0	0	2	1	0.56	Chicken
	0	0	1	0	0	5	0	0	1	2	0.56	Chips
	0	0	0	0	0	2	1	0	2	4	0.11	Cookie
	1	3	0	4	0	0	0	1	0	0	0.11	Lettuce
	1	0	0	2	0	1	0	0	3	2	0.33	Fries
	0	0	0	2	0	0	0	0	0	7	0.78	Pizza
	0.4	0.22	0.5	0.25	0.71	0.42	0.5	0.5	0.30	0.33	0.36	
	Predicted class											

(c) SVM output

Table 5.11. *Confusion Matrix with various classification algorithms with additional time domain features*

As the result, when the time domain parameters are added, it improves the result but it is not significant. The reason is that there are many parameters to classify. So three variables are not enough to make a significant improvement.

On the contrary, when the parameters are applied to decision theory, it yields a better result than the previous one . Tables 5.12, 5.13 and 5.14 show the result.

36	9	0.8	Crispy
9	36	0.8	Non crispy
0.8	0.8	0.8	precision
Predicted class			

(a) Quiet environment recording data

40	5	88.9	Crispy
9	36	80	Non crispy
81.6	87.8	84.4	precision
Predicted class			

(b) Noisy environment recording data

Table 5.12. *Confusion matrix with K-NN classifier for first level of decision tree*

35	11	76.1	Crispy
10	34	77.3	Non crispy
77.8	75.6	76.7	precision
Predicted class			

(a) Quiet environment recording data

40	8	83.3	Crispy
5	37	88.1	Non crispy
88.9	82.2	85.6	Precision
Predicted class			

(b) Noisy environment recording data

Table 5.13. *Confusion matrix with Neural Network classifier for first level of decision tree*

41	4	91.1	Crispy
15	30	66.7	Non crispy
73.2	88.2	78.9	Precision
Predicted class			

(a) Quiet environment recording data

43	2	95.6	Crispy
18	27	60	Non crispy
84.3	93.1	77.8	Precision
Predicted class			

(b) Noisy environment recording data

Table 5.14. *Confusion matrix with SVM classifier for first level of decision tree*

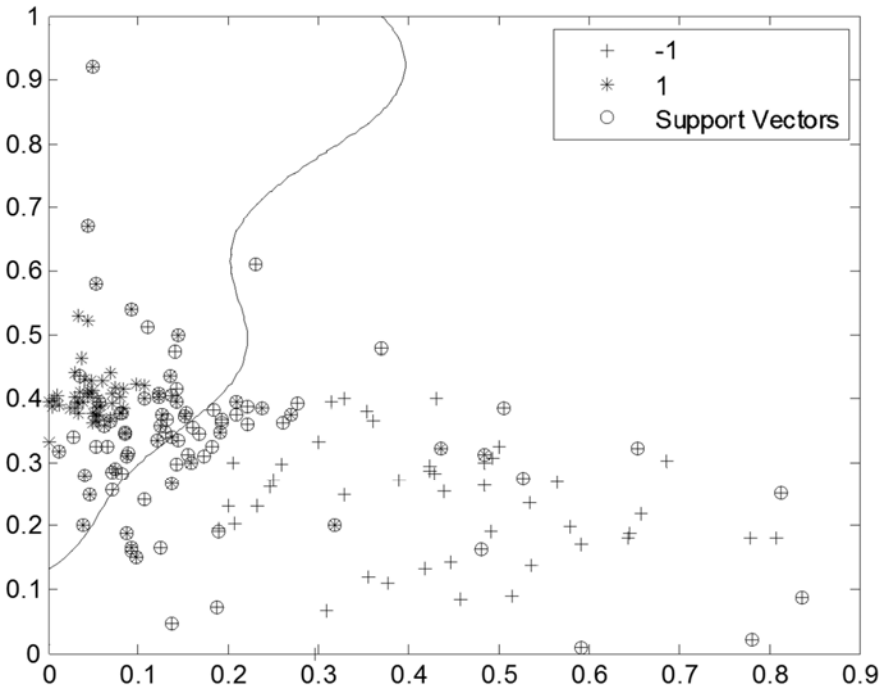


Figure 5.17. *Graphical output of SVM classification with RBF kernel for recorded data*

The result shows that the overall accuracy of first node is about 80%. Table 5.14 shows classification and identification with previous selected data on level-two.

Tables 5.15 and 5.17 and figure 5.20 represent the identification result at end branches. Three methods are used for each end branch. The application of the neural network requires some training parameters. Table 5.16 shows the summary of parameters and figure 5.18 shows the stop training condition with validation data.

Actual class						
	1504	855	1135	924	34.0	Carrot
	651	1577	935	672	41.1	Chips
	901	893	1111	1116	27.6	Cookie
	696	601	1124	1317	35.2	Lettuce
	40.1	40.2	25.8	32.7	34.4	Overall
	Predicted class					

(a) Classified in crispy food data

Actual class					
2537	884	1300	53.7	Burger
679	554	1154	23.2	Cereal
1168	1752	2005	40.7	Pizza
57.9	17.4	45.0	42.4	Overall
Predicted class				

(b) Classified in mixed food data

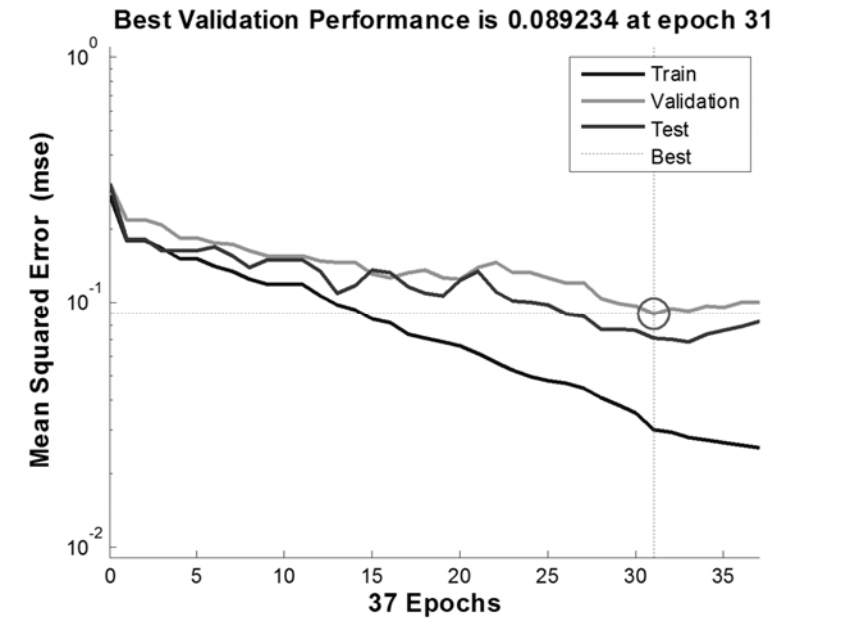
Actual class					
1303	1269	1562	31.5	Bread
1496	1760	1543	36.7	Chicken
1299	1259	1347	34.5	Fries
31.8	41.0	30.3	34.4	Overall
Predicted class				

(c) Classified in soft food data

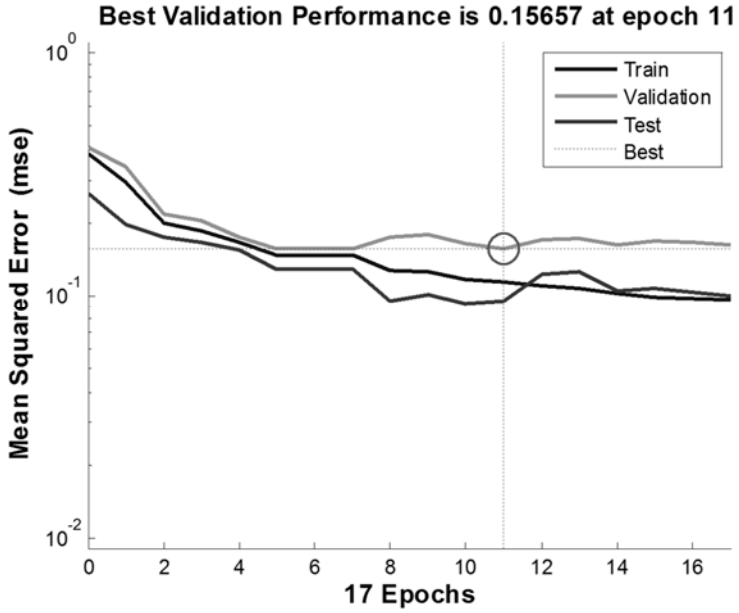
Table 5.15. *Confusion matrix with K-NN classifier for second level of decision tree. The data are classified by five groups which are crispy, mixed and soft chewing textures*

Number of	
Hidden layer	18
Input parameters	20
Output (Class)	5
Training sample	68
Validation sample	22
Testing sample	46/44

Table 5.16. *Parameters for neural network classifier*



(a) Stop condition in crispy data

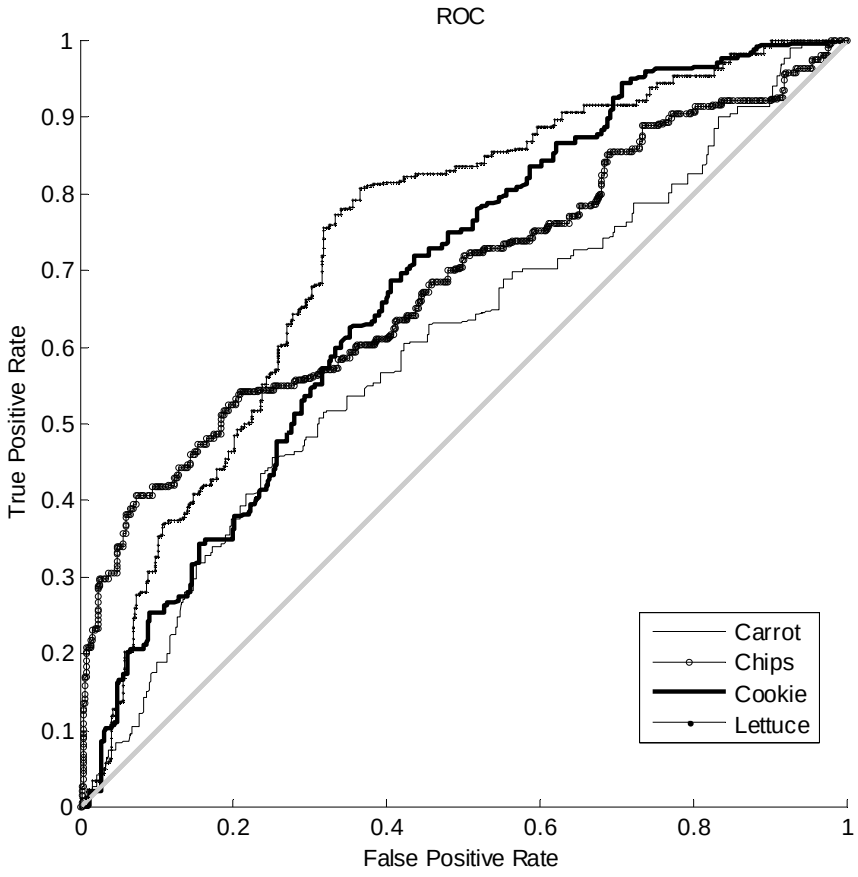


(b) Stop condition in noncrispy data

Figure 5.18. *Stop conditions of applying neural network in crispy and noncrispy data*

Actual class	1877	851	839	851	42.5	Carrot
	1179	1561	652	443	40.7	Chips
	1089	481	1405	1046	34.9	Cookie
	625	456	1192	1465	39.2	Lettuce
	39.4	46.6	34.4	38.5	39.4	Overall
	Predicted class					

(a) Confusion matrix for crispy data

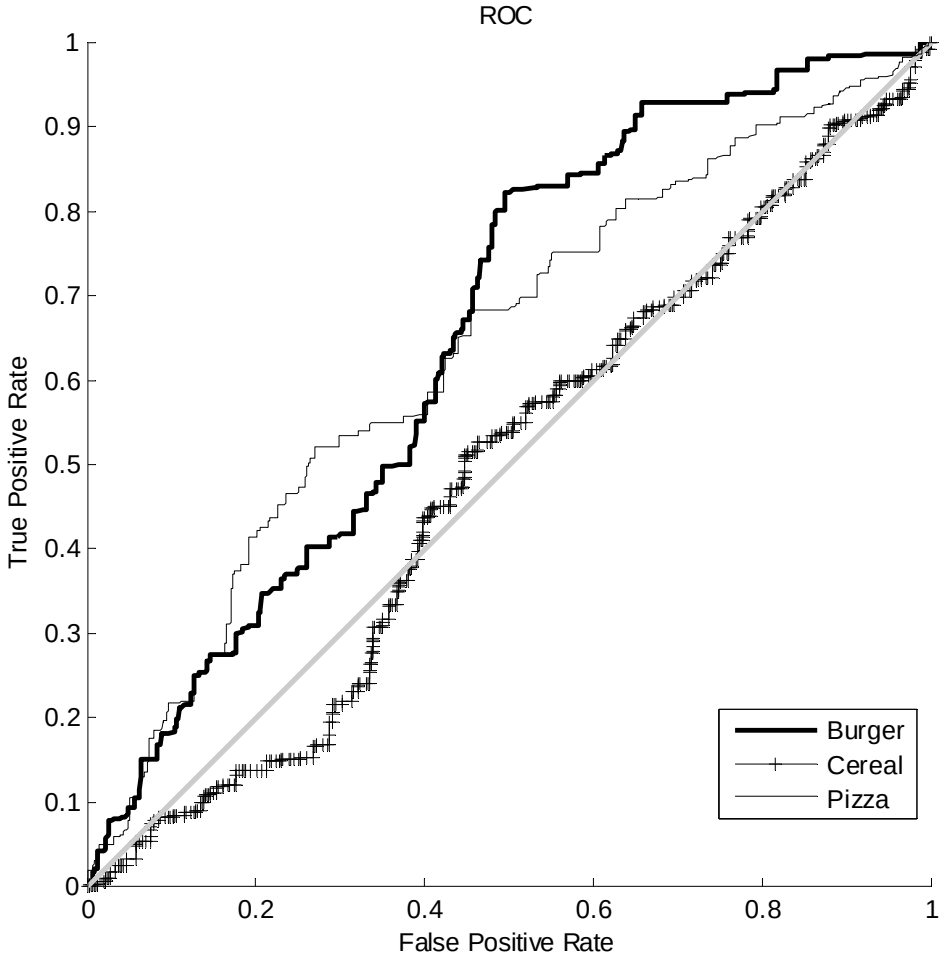


(b) ROC for crispy food data

Figure 5.19. *Classification using a neural network for crispy food data*

Actual class	3495	476	750	74.0	Burger
	1296	282	959	11.1	Cereal
	1997	813	2115	42.9	Pizza
	51.5	18.0	55.3	48.4	Overall
	Predicted class				

(a) Confusion matrix for mixed food data

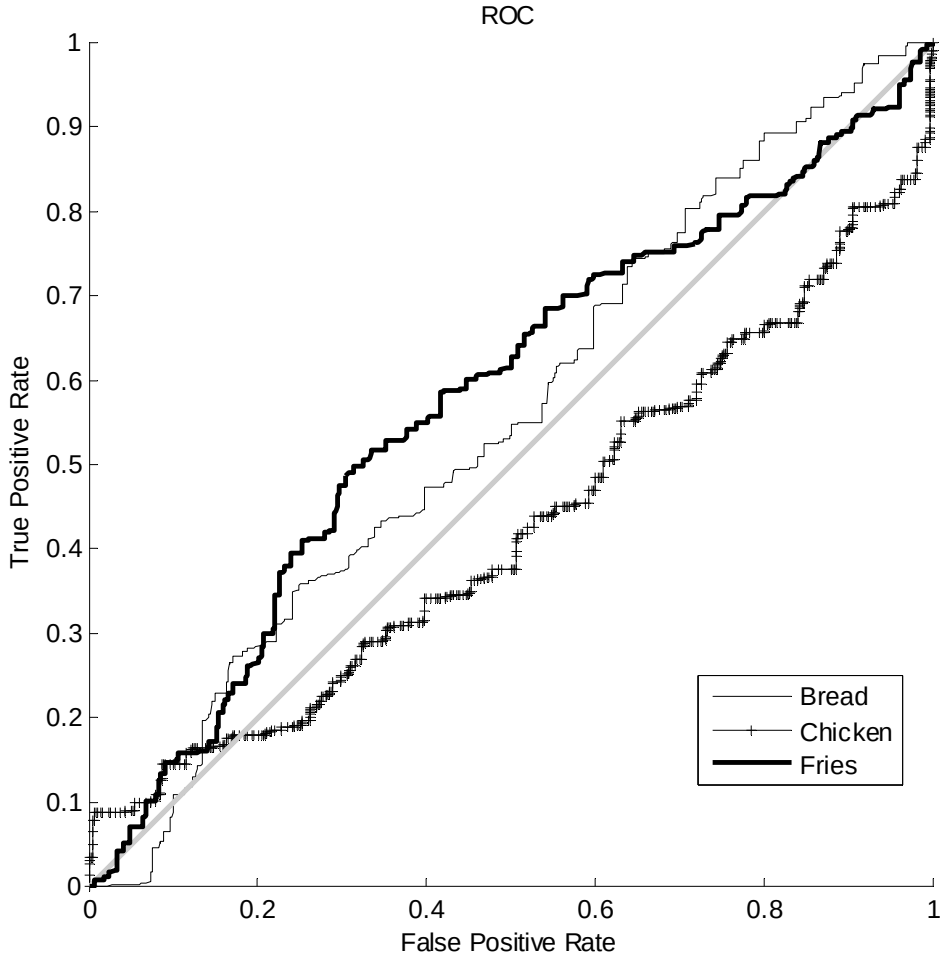


(b) ROC for mixed food data

Figure 5.20. *Classification with a neural network for mixed food data*

Actual class	1358	1882	894	32.8	Bread
	1599	2098	1102	43.7	Chicken
	1173	1627	1105	28.3	Fries
	32.9	37.4	35.6	35.5	Overall
	Predicted class				

(a) Confusion matrix for soft food data



(b) ROC for soft food data

Figure 5.21. *Classification with a neural network for soft food data*

Actual class	1939	949	804	726	43.9	Carrot
	1070	1555	665	545	40.5	Chips
	993	729	1366	933	34.0	Cookie
	714	583	1101	1340	35.8	Lettuce
	41.1	40.7	34.7	37.8	38.7	Overall
	Predicted class					

(a) Classified in crispy food data

Actual class	3086	826	809	65.4	Burger
	1064	440	883	18.4	Cereal
	1912	1230	1783	36.2	Pizza
	50.9	17.6	51.3	44.1	Overall
	Predicted class				

(b) Classified in mixed food data

Actual class	889	1630	1615	21.5	Bread
	1345	1974	1480	41.1	Chicken
	1183	1442	1280	32.8	Fries
	26.0	39.1	29.3	32.3	Overall
	Predicted class				

(c) Classified in soft food data

Table 5.17. *Confusion matrix of identification rate by SVM with crispy, mixed and soft food data*

From the previous selected data, the overall accuracy with KNN is 34%, 42% and 34%, with Neural Network is 39%, 48% and 35% and with SVM is 38%, 44% and 32%. The result shows better rates than the previous which is compared with all classes.

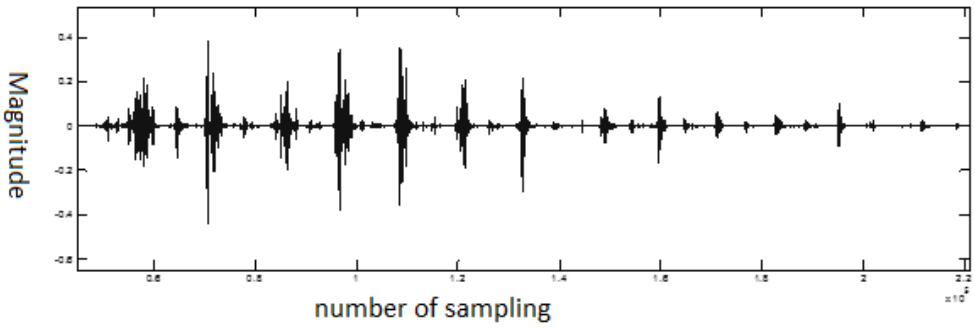
There are two more frequency features that weren't used for classification and identification. If those variables are used, it shows better performance.

Chapter 6

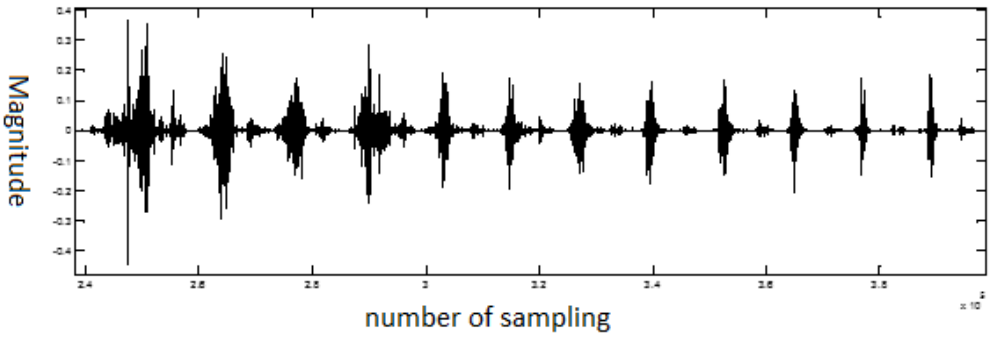
Bite weight estimation

6.1 Background

Amft [8] used multilinear regression for weight estimation. The algorithm used seven statistical parameters from the chewing sounds. The accuracy was about 60%. We added several more parameters. One of the parameters was acquired from the quantity of food for each intake. There are two ways to eat more food in the same time. One is speed. The other is quantity. But Fig. 6.1 shows that a larger bite of food (chips) yields a larger amplitude. Examining the time domain shows that the signal persists longer for the increased amount.



(a) Chewing sounds for one chip in the mouth



(b) Chewing sounds for 2 or 3 chips in the mouth

Figure 6.1 *Chewing sounds for one or more potato chips in the mouth*

Figure 6.1 shows that it is difficult to make a model to use the statistical parameters. Figure 6.2 shows that if the food intake is increased, the signal duration and sound energy don't change. Thus this case requires different parameters that Amft didn't suggest. We developed statistical parameters of quantity.

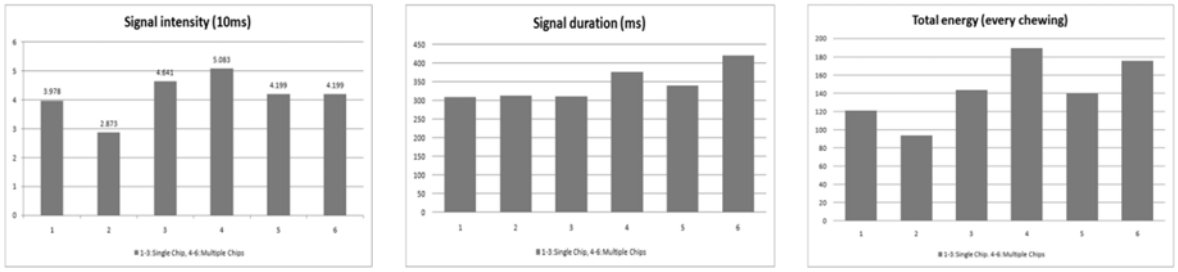


Figure 6.2. *Signal intensity, duration and signal energy for different amounts of food acquired during a fixed duration of between 300 and 430 ms.*

6.2 Experimental setup

For the bite weight estimation experiment, two subjects were selected and fed several different amount of potato chips. The reason to select potato chips is that the event pattern in the time domain was easy to identify and measure. There were six potato chip groups, 2, 4, 5, 8, 10 and 12 g served to each subject. The same smartphone based recording system recorded the data.

Three parameters, number of chewing events, total duration of chewing events and total energy of chewing events were selected for estimation.

The regression method was used to estimate bite weight. Also, there were two linear regression methods such as single linear regression and multilinear regression. In this research, both regression methods were used.

Regression analysis is the statistical technique for estimating the relationships among variables.

General regression theory was established by K. Pearson. He gathered more than 1000 physical data from families and determined the relationship between fathers' and sons' heights.

In regression analysis, there are two variables and the most important variable is called the independent variable and the other is called the dependent variable.

It originally started for genetic research but it is widely used in all scientific and engineering fields.

The regression analysis is defined by a statistical equation between dependent and independent variables. However, to estimate the relationship between two variables, the function has to be defined from the equation in two variables.

These two variables can make a plot which is called a scatter plot. To estimate the function, the first step is plotting the scatter plot. The scatter plot shows the relationship level easily.

From the scatter plot, we can acquire three important points.

1. The relationship between the two variables is negative or positive.
2. It is linear or nonlinear
3. The level of the relationship

From the scatter plot, we can determine the regression model. The model can be linear or nonlinear. If the model is linear, it is defined by the equation,

$$Y = b_0 + b_1 X$$

The correlation coefficient is called R^2 is very useful to define the model. R^2 is defined by the equation,

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - y)^2}$$

When R^2 is close to 1, the model is adapted to the data. If $R^2$is close to 0, the regression model is not related to the data.

Multiple regression analysis is an extended model of single regression analysis and it has two or more independent variables. To define dependent variable *Y*, if it requires *k* number of independent variables, it is called a multiple regression model and is represented by equation,

$$Y = b_0 + b_1X_1 + b_2X_2 \cdots b_nX_n$$

Single linear regression consists of one independent and one dependent variable. But, in general, it is uncommon to build an equation using a single variable. Most problems are complex and use multiple variables.

In this research, I want to estimate bite weight. So the dependent variable is weight and independent variables are number of chewing events, total duration of chewing events and total energy of chewing events.

The first analysis assumed the linear regression model and used one independent variable, the number of chewing events. From the scatter plot, we determined the relationship between the two variables.

6.3 Estimation

Single linear regression

As mentioned in the background information for regression, the first step is drawing a scatter plot. Figure 6.3 shows the scatter plot with sample #1. The independent variable is weight and the dependent variable is number of chewing events. These two variables have a good linear positive relationship. So the data are good to make a linear regression model.

Figure 6.4 represents the linear regression model. To find out the quality of regression model, R^2 is evaluated. Table 6.1 shows the R^2 , which is close to 1. So it is well modeled.

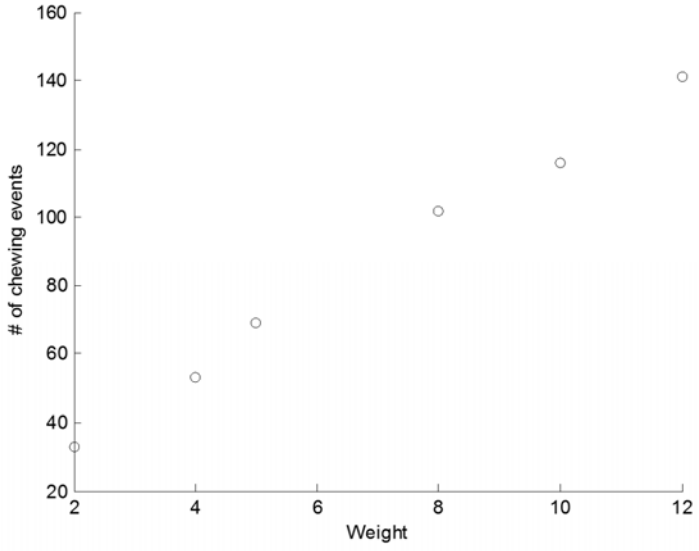


Figure 6.3. *Scatter plot for bite weight experiment of subject #1*

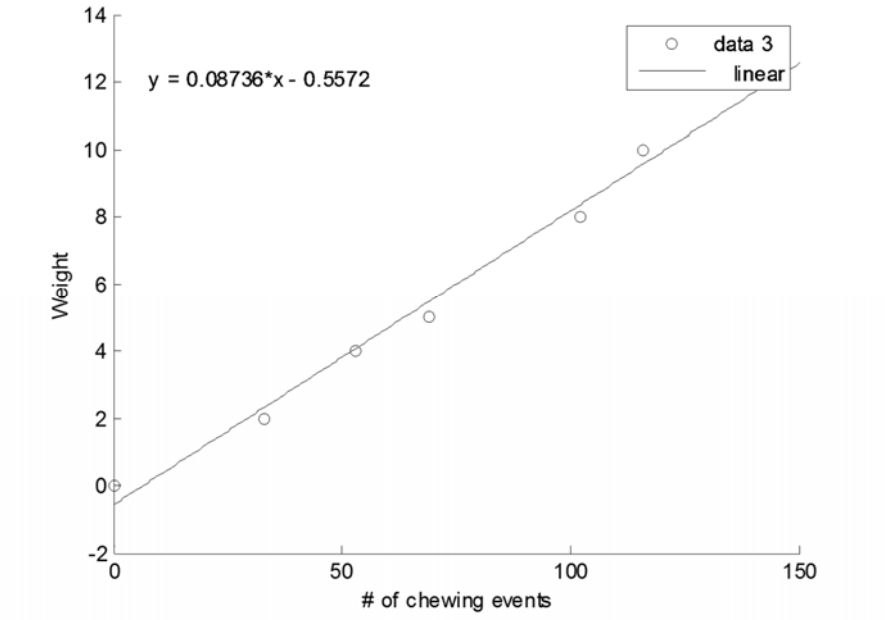


Figure 6.4. *Linear regression model with bite weight data*

Experiment #1: $Y = 0.08736X - 0.5572$

Experiment #2: $Y = 0.1345X - 0.6172$

R	R Square	Std. Error of the Estimate
0.997[a]	0.995	0.30429

Table 6.1. R^2*for single regression*

Multiple linear regression

For the multiple linear regression, two more individual variables, duration of chewing events and energy of chewing events, were added.

Both multiple linear regression models eliminate X_2 (total duration of chewing events). The reason is the correlation of $X_2$with other variables is around one. Table 6.2 shows R^2 for the quality of the regression model.

Experiment #1: $Y = -0.166 + 0.169X_1 - 0.0001X_3$

Experiment #2: $Y = -0.3619 + 0.448X_1 + 0.0005X_3$

X_1: Number of chewing events

X_2: Total duration of chewing events

X_3: Total energy of chewing events

R	*R* Square	Std. Error of the Estimate
1.000[a]	1.000	0.13132

Table 6.2. *R^2 for multiple regression*

Estimation of calories

The previous chapter presents food identity and an equation of bite weight estimation. Application of this information into the data for 12 subjects yielded an estimation of bite weight and calories.

From figure A.1, Potato chips contain 160 calories for 28 g.

Subject	1	2	3	4	5	6	7	8	9	10	11	12
Chewing Events #1	0.55	1.17	1.52	0.94	0.08	0.75	1.13	0.64	1.72	1.45	0.46	0.84
Chewing Events #2	0.7	1.80	1.14	0.99	0.01	0.61	1.20	0.29	1.86	1.54	0.62	1.46
Chewing Events #3	0.89	1.61	1.74	0.92	0.38	1.11	1.33	0.81	1.29	1.44	0.60	0.80
Total Events	2.17	4.5	4.4	2.86	0.31	2.48	3.66	1.75	4.88	4.43	1.69	3.12
	12.37	25.65	25.08	16.30	1.77	14.14	20.86	9.98	27.82	25.25	9.63	17.78

Table 6.3. *Estimated weight by single regression analysis*

Subject	1	2	3	4	5	6	7	8	9	10	11	12
Chewing Events #1	0.86	2.07	1.66	0.59	0.59	0.45	0.99	0.86	1.53	1.13	0.72	0.86
Chewing Events #2	0.72	1.80	1.80	0.59	0.18	0.45	0.99	0.59	2.07	1.13	0.86	1.13
Chewing Events #3	1.40	1.53	2.61	0.45	0.59	1.13	0.86	1.13	0.99	1.26	0.86	0.72
Total Events	2.99	5.41	6.08	1.64	1.37	2.04	2.85	2.58	4.60	3.52	2.45	2.72
	17.04	30.84	34.66	9.35	7.81	11.63	16.25	14.71	26.22	20.06	13.97	15.50

Table 6.4. *Estimated weight by multiple regression analysis*

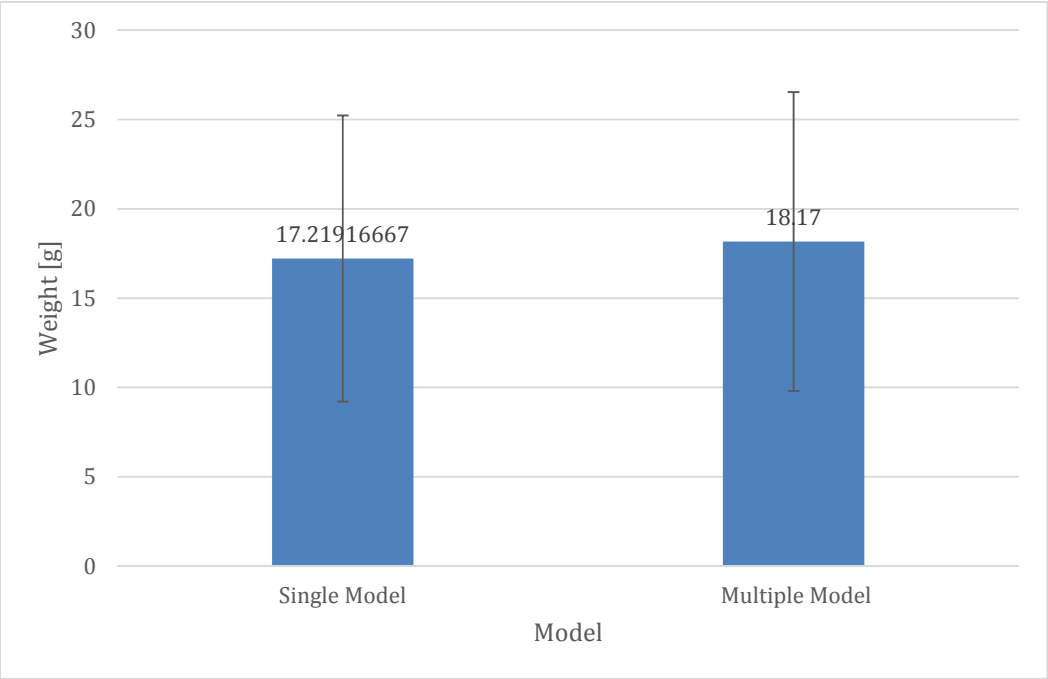


Figure 6.5 *Estimated weight by single and multiple linear regress analysis*

Table 6.3 shows weight estimation using single linear regression. Table 6.4 shows weight estimation using multiple linear regression. If two estimated values with single and multiple are similar, the regression model is fit and it can estimate well.

Subject #9 consumes about 26~28 calories from potato chips.

The average values of estimated weights are 17.2 g and 18.2 g. Their standard deviations are 8 and 8.4. The estimated weight varies from 80% to 90%. Two methods to decrease errors are reducing bias and variance. Making a new regression model for every subject can reduce the bias. Examining the regression analysis, the single and multiple linear regression analysis has two different equations. The two models are calculated for each experiment because there are bias errors in modeling. this is the reason why a new regression model is required for every subject. Control of variance is an alternative method to reduce the error. Unfortunately, because of lack of experimental data, it is impossible to analyze the variance. But in general, increasing the number of samples helps to decrease variance.

The estimation of calories directly from a single food such as potato chips is impossible because there are not enough data to estimate bite weight for all kinds of food. The research has only one model, potato chips, which is applied to estimate the weight. Besides, the model is applied to two subjects and two subjects is not enough to perform evaluation for different subjects. Another problem is that the accuracy and its related values such as recall and precision of the classification is low. For example, for potato chips, the recall value is only 56% (Table 5.15). It means 56% potato chips chewing sound can be identified as chips. So 44% of the data are missing to compute the weight. But, on the other hand, if there is a constraint condition, it is possible to determine calories. The first condition is that we should know the consumed food clearly. In this condition, we can apply regression model for all chewing events and get estimated values. However, if we don't know the consumed food clearly, the calculation follows as I described before. The second condition is eating habits.

Before estimation, we assume the habit is quite regular and the variance is enough small to ignore. So the missing 44% of the data can be estimated by 56% of chewing signal. If we don't know the consumed food clearly, estimating calories by single food consumption is impossible.

Estimation of calories from multiple foods is more complex. The estimation is ruled by precision values. From Table 5.15, the precision of potato chips is 62.5%. The remaining 25% comes from cereal with milk and 12.5% comes from lettuce. So 37.5% of the data are missing and coming from different sources. If calories of cereal with milk and lettuce are similar to potato chips, the estimation yields an approximate value. Figures A.1, A.2 and A.3 show the nutrition facts of three different foods. Potato chips has 5.7 cal/g, cereal with milk has 5 cal/g and iceberg lettuce has 0.11cal/g. Cereal with milk has similar calories but lettuce is quite different. So the estimation errors are 12.5% for lettuce and a small portion of 25% of cereal with milk.

To make these steps simpler, the following procedure shows how to calculate estimated calories.

1. Estimate type of food
2. Estimate quantity of food.

2.1 Calibrate the regression model for every subject

2.2 Calibrate the variance model for every subject

3. Use a multiple regression model, to calculate weight
4. Go to step 2 until all sample foods are included.
5. Estimate calories by weight information and nutrition facts

Chapter 7

Summary and conclusions

- Chapter 2 presents the experimental setup such as food, subject selection and recording device. There were 12 subjects who didn't have any allergies or jaw problems who participated in the research. Foods used were 10 solid foods and 3 drinks based on the two previous surveys. The smart phone based recording has many advantages due to its portability and performance. Because the microphone shape is the same as a head set, it preserves the patients' privacy.
- Chapter 3 shows segmentations by swallowing and by chewing events in one cycle of swallowing. For swallowing, segmentation works for drinks but not for solid food because the intensity of the swallowing signal is smaller than that for the chewing signals. In chewing event segmentation, the new algorithm uses envelopes of signals and their peaks. The performance of the classic algorithm (Otsu's) shows 92% accuracy but the new algorithm shows 98% accuracy for all sound data.
- Chapter 4 extracts several features from the chewing sounds. For group parameters, LPC and MFCC were applied. For a single parameter split frequency domain and time domain were applied.
- Chapter 5 represents three classification/identification algorithms such as K-NN, Neural Network and SVM. Their performance was similar and low (15~25%) when all features are used for classification. I suggested a hybrid hierarchy classification method with a decision tree. On the first level of the tree, three time domain features were used and showed more than 75% accuracy. Multiple classification was

evaluated by previously classified data for the second level. Their accuracies were different but there was a classification improvement (25~40%).

- Chapter 6 used bite weight estimation. Single and multiple linear regression methods were applied to make an equation for weight and other variables such as sound energy and time. It is well modeled but because of the subjects' different properties, one estimated value shows close estimated weight from single and multiple regression model.

In conclusion, I've used a two segmentation model, three classification methods and one new hybrid classification method. In segmentation, the performance was high (92% ~ 98%). So most of the chewing events can be included for the next step of analysis. LPC and MFCC can't distinguish features as shown by previous research or other types of research. The first reason is that food does not have constant properties. During the food preparation step, although it is same food, there can be different chewing textures from different cooking conditions. The second reason is mixed ingredient foods. The food consists of several ingredients and the chewing sounds can be changed by bite location. The third reason is related to subjects' properties. Even if they eat the same food, they can make different sounds. For classification methods, increasing parameters doesn't improve the accuracy. The more important point is making distinguishable features. The hybrid method using the decision tree improves the performance. Within a food group that has similar chewing textures, it shows a better result. Also, as mentioned in previous research, crispy texture can yield a better identification. In weight estimation, although the model is well estimated ($R^2 = 0.995, R^2 = 1$), the weights have variations and two weights which are acquired by single

and multiple regression models are different. This is because people vary. From tables 6.3 and 6.4, the regression model is fit and it can estimate well.

There were two experiments for estimating bite weight. The results show two different regression models by two subjects. Modeling for every subject can decrease the bias of the error. But in this paper, there is no way to investigate the variance due to lack of experimental data. But increasing the number of samples could reduce the variance generally.

For a single item, if it is potato chips, the best algorithm (K-NN) has 56% of recall value. It can calculate only 56% of calories from the data and remaining 44% is missing.

Estimations from ten different foods show different results because some data comes from different types of foods. When potato chips are applied, 25% and 12.5% of the data come from cookies and lettuce. So 37.5% of the data comes from wrong sources and 62.5% of the data comes from right source for the precision of potato chips in Table 5.15,. As described at the end of the previous chapter, it is impossible to calculate calories directly. It requires some constraint conditions.

7.1 Future direction

The goal of this research is to estimate calories using chewing sounds. First we estimate the type of food. Second we estimate the quantity of food, then we estimate calories. For the case of potato chips, the standard deviation is high and can't be ignored.

Future work will be focused on decreasing the estimation error. There are two points to reduce the error.

1. Improving accuracy of classification method

- I used the cross validation method for classification in this research. To improve and reduce bias, self-validation will be required. So all subjects must prerecord their sound patterns for all kinds of food and it will be helpful for increasing accuracy.
- Generally, increasing sample number is good for decreasing variance in error.
- Finding new distinct features in sound pattern is required. I didn't apply peak frequency as chewing events were added. This will be one of helpful features.

2. Improving bite weight estimation

- Improving bite weight estimation requires making a new multiple linear regression model, and calibration for all subjects. This should decrease bias error.
- There is no deep investigation of variance in this research. So we need to acquire more data for variance research. All subjects should repeat the bite weight experiment several times to yield variance data.

In addition, we need to improve the tree structure and to clarify the several ingredients of food. The recording application in the smartphone has a basic recording function. An additional function such as taking pictures and recording on the memo pad would help to improve the accuracy of identification.

APPENDIX A. Source Code

R02_DataSegment

```
%% Bread

[R02_Bread1_Seg, R02_Bread1_count] = Segmentation(R02_Bread_1, 1000,
44100, 'Subject#2 Bread cycle#1', 1);
[R02_Bread2_Seg, R02_Bread2_count] = Segmentation(R02_Bread_2, 1000,
44100, 'Subject#2 Bread cycle#2', 1);
[R02_Bread3_Seg, R02_Bread3_count] = Segmentation(R02_Bread_3, 1000,
44100, 'Subject#2 Bread cycle#3', 1);

for i=1:R02_Bread1_count
    if i == 1
        DR02_Bread1_Segment =
DR02_Bread_1(R02_Bread1_Seg(1,i):R02_Bread1_Seg(2,i));
    else
        DR02_Bread1_Segment = [DR02_Bread1_Segment;
DR02_Bread_1(R02_Bread1_Seg(1,i):R02_Bread1_Seg(2,i))];
    end
end

for i=1:R02_Bread2_count
    if i == 1
        DR02_Bread2_Segment =
DR02_Bread_2(R02_Bread2_Seg(1,i):R02_Bread2_Seg(2,i));
    else
        DR02_Bread2_Segment = [DR02_Bread2_Segment;
DR02_Bread_2(R02_Bread2_Seg(1,i):R02_Bread2_Seg(2,i))];
    end
end

for i=1:R02_Bread3_count
    if i == 1
        DR02_Bread3_Segment =
DR02_Bread_3(R02_Bread3_Seg(1,i):R02_Bread3_Seg(2,i));
    else
        DR02_Bread3_Segment = [DR02_Bread3_Segment;
DR02_Bread_3(R02_Bread3_Seg(1,i):R02_Bread3_Seg(2,i))];
    end
end

R02_Bread1_Segment = miraudio(DR02_Bread1_Segment);
R02_Bread2_Segment = miraudio(DR02_Bread2_Segment);
R02_Bread3_Segment = miraudio(DR02_Bread3_Segment);
```

R01_GetSNR.m

```
%% Variable

%% Bread
for i=1:R03_Bread1_count(1)
    if i == 1
        DR03_Bread1_Noise = DR03_Bread_1(1:R03_Bread1_Seg(1,i));
    else
        DR03_Bread1_Noise = [DR03_Bread1_Noise;
DR03_Bread_1(R03_Bread1_Seg(2,i-1):R03_Bread1_Seg(1,i))];
        if i==R03_Bread1_count(1)
            DR03_Bread1_Noise = [DR03_Bread1_Noise;
DR03_Bread_1(R03_Bread1_Seg(2,i):length(DR03_Bread_1))];
        end
    end
end

for i=1:R03_Bread2_count
    if i == 1
        DR03_Bread2_Noise = DR03_Bread_2(1:R03_Bread2_Seg(1,i));
    else
        DR03_Bread2_Noise = [DR03_Bread2_Noise;
DR03_Bread_2(R03_Bread2_Seg(2,i-1):R03_Bread2_Seg(1,i))];
        if i==R03_Bread2_count(1)
            DR03_Bread2_Noise = [DR03_Bread2_Noise;
DR03_Bread_2(R03_Bread2_Seg(2,i):length(DR03_Bread_2))];
        end
    end
end

for i=1:R03_Bread3_count
    if i == 1
        DR03_Bread3_Noise = DR03_Bread_2(1:R03_Bread2_Seg(1,i));
    else
        DR03_Bread3_Noise = [DR03_Bread3_Noise;
DR03_Bread_3(R03_Bread3_Seg(2,i-1):R03_Bread3_Seg(1,i))];
        if i==R03_Bread3_count(1)
            DR03_Bread3_Noise = [DR03_Bread3_Noise;
DR03_Bread_3(R03_Bread3_Seg(2,i):length(DR03_Bread_3))];
        end
    end
end

a =
mirgetdata(mirspectrum(miraudio(DR03_Bread1_Segment),'Length',8192));
b =
mirgetdata(mirspectrum(miraudio(DR03_Bread1_Noise),'Length',8192));
pa = sum(a(1:1024));
pb = sum(b(1:1024));
```

```
snr = 10*log10((pa-pb)/pb);
R03_SNR(1,1) = snr;

a =
mirgetdata(mirspectrum(miraudio(DR03_Bread2_Segment),'Length',8192));
b =
mirgetdata(mirspectrum(miraudio(DR03_Bread2_Noise),'Length',8192));
pa = sum(a(1:1024));
pb = sum(b(1:1024));
snr = 10*log10((pa-pb)/pb);
R03_SNR(2,1) = snr;

a =
mirgetdata(mirspectrum(miraudio(DR03_Bread3_Segment),'Length',8192));
b =
mirgetdata(mirspectrum(miraudio(DR03_Bread3_Noise),'Length',8192));
pa = sum(a(1:1024));
pb = sum(b(1:1024));
snr = 10*log10((pa-pb)/pb);
R03_SNR(3,1) = snr;
```

Parameterization.m

```
function [do_mfcc, o_bright, o_center, o_subband, o_power,
o_bandwidth, o_pitch] = Parameterization(i_segment,mfcc_rank)
% MFCC
o_mfcc = mirmfcc(i_segment,'Rank',1:mfcc_rank);
do_mfcc = mirgetdata(o_mfcc);
% Brightness (Bread1~3)
o_bright = mirgetdata(mirbrightness(i_segment));
% Central frequency (Bread1~3)
o_center = mirgetdata(mircentroid(mirspectrum(i_segment)));
% Subband Power (Bread1)
o_spectrum = mirgetdata(mirspectrum(i_segment));
o_subband = zeros(1,3);
for i=1:round(length(o_spectrum)/8)
    o_subband(1) = o_subband(1) + power(o_spectrum(i),2);
end
o_subband(1) = log(o_subband(1));
for i=round(length(o_spectrum)/8)+1:round(length(o_spectrum)/4)
    o_subband(2) = o_subband(2) + power(o_spectrum(i),2);
end
o_subband(2) = log(o_subband(2));
for i=round(length(o_spectrum)/4)+1:round(length(o_spectrum)/2)
    o_subband(3) = o_subband(3) + power(o_spectrum(i),2);
end
o_subband(3) = log(o_subband(3));
% Power Spectrum
o_power = 0;
for i=1:length(o_spectrum);
    o_power = o_power + power(o_spectrum(i),2);
```

```
end
o_power = log(o_power);
% Bandwidth
temp=0;
temp1=0;
for i=1:length(o_spectrum)
    temp = temp+(power((22050/length(o_spectrum))-o_center,
2)*power(o_spectrum(i),2));
    temp1 = temp1+power(o_spectrum(i),2);
end
o_bandwidth = sqrt(temp/temp1);
% Pitch frequency
o_pitch = mirgetdata(mirpitch(i_segment, 'Mono'));
```

Segmentation.m

```
function [segmentation_set, segmentation_length] =
Segmentation(mir_wave, Peak_Distance, Sampling_Rate, TITLE, env_on)

% mir_wave
ori_wave = mirgetdata(mir_wave);
envelope_wav = mirenvelope(mir_wave);
env_x = 0:1/Sampling_Rate:(length(mirgetdata(envelope_wav))-
1)/Sampling_Rate;
if env_on == 1
    figure(1);
    plot(env_x, mirgetdata(envelope_wav),'k');
end
envelope_wav = mirgetdata(envelope_wav);
Count_mir_wave = length(envelope_wav) * 0.67;
[hn,hx]=hist(envelope_wav, 512);
total_hist = 0;
count = 1;
while total_hist < Count_mir_wave
    total_hist = total_hist + hn(count);
    count=count+1;
end
threshold = hx(count);

% x = 0:1/44100:(length(envelope_wav)-1)/44100;
% plot(x, envelope_wav,'b',x,threshold,'r');
Max_wav = max(ori_wave);
Min_wav = min(ori_wave);
Max_envelope_wav = max(envelope_wav);
Min_envelope_wav = min(envelope_wav);
Mean_envelope_wav = mean(envelope_wav);
Median_envelope_wav = median(envelope_wav);

% Finding starting, end & peak point

Start_mir_wave = zeros(1,50);
```

```
End_mir_wave = zeros(1,50);
Peak_mir_wave = zeros(1,500);
count_start = 1;
count_end = 1;
count_peak = 1;

for i=3:length(envelope_wav)-4
    if envelope_wav(i) < threshold
        if envelope_wav(i+1) > threshold
            Start_mir_wave(count_start) = i*16;
            count_start = count_start + 1;
        end
    end

    if envelope_wav(i) > threshold
        if envelope_wav(i+1) < threshold
            End_mir_wave(count_end) = i*16;
            count_end = count_end + 1;
        end
    end
end

[pks, Peak_mir_wave] = findpeaks(mirgetdata(mirenvelope(mir_wave)),
'MINPEAKDISTANCE', Peak_Distance);

for i=1:length(Peak_mir_wave)
    temp_peak = Peak_mir_wave(i);
    if env_on == 1
        line([temp_peak/Sampling_Rate, temp_peak/Sampling_Rate],
[0,Max_envelope_wav], 'color', 'r', 'linestyle', '-');
    end
    Peak_mir_wave(i) = Peak_mir_wave(i) * 16;
end

% Matching Start, End & Peak points
% Peak point has to place between start and end points

Temp_Set_mir_wave = zeros(4, 100);
temp_count_set = 1;
for i=1:length(Peak_mir_wave)
    Close_Start_mir_wave = 100000000;
    Close_End_mir_wave = 100000000;
    Temp_Set_mir_wave(3, temp_count_set) = Peak_mir_wave(i);
    Temp_Set_mir_wave(4, temp_count_set) = pks(i);
    for j=1:count_start-1
        if Peak_mir_wave(i)-Start_mir_wave(j) > 0
            if Close_Start_mir_wave > abs(Peak_mir_wave(i)-
Start_mir_wave(j))
                Close_Start_mir_wave = abs(Peak_mir_wave(i)-
Start_mir_wave(j));
```

```
                Temp_Set_mir_wave(1, temp_count_set) =
Start_mir_wave(j);
            end
        end
    end
    for k=1:count_end-1
        if End_mir_wave(k)-Peak_mir_wave(i) > 0
            if Close_End_mir_wave > abs(End_mir_wave(k)-
Peak_mir_wave(i))
                Close_End_mir_wave = abs(Peak_mir_wave(i)-
End_mir_wave(k));
                Temp_Set_mir_wave(2, temp_count_set) =
End_mir_wave(k);
            end
        end
    end
    temp_count_set = temp_count_set + 1;
end

Set_mir_wave = zeros(3, 50);
count_set = 1;
Min_interval_threshold = 2000;
Max_interval_threshold = 40000;

for i=1:temp_count_set-1
    if abs(Temp_Set_mir_wave(2, i) - Temp_Set_mir_wave(1,i)) >
Min_interval_threshold
    if abs(Temp_Set_mir_wave(2, i) - Temp_Set_mir_wave(1,i)) <
Max_interval_threshold
        if count_set > 1
            if Set_mir_wave(1, count_set-1) ~= Temp_Set_mir_wave(1,i)
                if Temp_Set_mir_wave(2,i) ~= 0
                    if Temp_Set_mir_wave(1,i) ~= 0
                        Set_mir_wave(1, count_set) =
Temp_Set_mir_wave(1,i);
                        Set_mir_wave(2, count_set) =
Temp_Set_mir_wave(2,i);
                        Set_mir_wave(3, count_set) =
Temp_Set_mir_wave(3,i);
                        Set_mir_wave(4, count_set) =
Temp_Set_mir_wave(4,i);
                        count_set = count_set + 1;
                    end
                end
            end
        else
            if Temp_Set_mir_wave(2,i) ~= 0
                if Temp_Set_mir_wave(1,i) ~= 0
                    Set_mir_wave(1, count_set) =
Temp_Set_mir_wave(1,i);
```

```
                    Set_mir_wave(2, count_set) =
Temp_Set_mir_wave(2,i);
                    Set_mir_wave(3, count_set) =
Temp_Set_mir_wave(3,i);
                    Set_mir_wave(4, count_set) =
Temp_Set_mir_wave(4,i);
                    count_set = count_set + 1;
                end
            end
        end
    end
    end
end

x = 0:1/Sampling_Rate:(length(ori_wave)-1)/Sampling_Rate;
figure(2);
plot(x, ori_wave,'k');
title(TITLE);
xlabel ('Time [sec]');

for i=1:count_set-1;
    line([Set_mir_wave(1,i)/Sampling_Rate,
Set_mir_wave(1,i)/Sampling_Rate], [Min_wav,Max_wav], 'color', 'k',
'linestyle', '-');
    line([Set_mir_wave(2,i)/Sampling_Rate,
Set_mir_wave(2,i)/Sampling_Rate], [Min_wav,Max_wav], 'color', 'k',
'linestyle', '-');
end

for i=1:count_set-1
    segmentation_set(1,i) = Set_mir_wave(1,i);
    segmentation_set(2,i) = Set_mir_wave(2,i);
    segmentation_set(3,i) = Set_mir_wave(3,i);
    segmentation_set(4,i) = Set_mir_wave(4,i);
end;
segmentation_length(1) = count_set-1;
segmentation_length(2) = Max_envelope_wav;
segmentation_length(3) = Min_envelope_wav;
segmentation_length(4) = Mean_envelope_wav;
segmentation_length(5) = std(segmentation_set(4,:));

% Test code
% for i=1:count_start-1
%     line([Start_mir_wave(i), Start_mir_wave(i)], [-1,1], 'color',
'r', 'linestyle', '-');
% end
% for i=1:length(Peak_mir_wave)
%     line([Peak_mir_wave(i), Peak_mir_wave(i)], [-1,1], 'color', 'r',
'linestyle', '-');
% end
% for i=1:count_end-1
```

```
%     line([End_mir_wave(i), End_mir_wave(i)], [-1,1], 'color', 'y',
'linestyle', '-');
% end
```

KNN_Classification.m

```
%% K-NearestNeighbor
%

clear;
clc;

% Training_Sample_dimension = 180;
% Testing_Sample_dimension = 90;
% Output_dimension = 10;

Training_Sample_dimension = 90;
Testing_Sample_dimension = 45;
Output_dimension = 5;

Element_dimension = 20;
NearestNeighbor = 16;
E_distance =
zeros(Testing_Sample_dimension,Training_Sample_dimension);
NN_matrix = zeros(2,Training_Sample_dimension);
Output_matrix = zeros(1,Output_dimension);
Testing_output_matrix = zeros(1,Testing_Sample_dimension);

Training_Sample = load('-ascii','KNN_Crispy_Training.txt');
Testing_Sample1 = load('-ascii','KNN_Crispy_Q_Test.txt');
Testing_Sample2 = load('-ascii','KNN_NTest_wEnergy.txt');

% Training_Sample = load('-
ascii','Segment_NoNoise_Crispy_KNN_Train.txt');
% Testing_Sample1 = load('-
ascii','Segment_NoNoise_Crispy_KNN_Test.txt');
% Testing_Sample2 = load('-
ascii','Segment_Noise_Crispy_KNN_Test.txt');

% Training_Sample = load('-
ascii','Segment_NoNoise_NoCrispy_KNN_Train.txt');
% Testing_Sample1 = load('-
ascii','Segment_NoNoise_NoCrispy_KNN_Test.txt');
% Testing_Sample2 = load('-
ascii','Segment_Noise_NoCrispy_KNN_Test.txt');

Target_output_matrix =
transpose(Testing_Sample1(:,Element_dimension+1));

for i=1:Testing_Sample_dimension
    E_distance(i) = 0;
end

for te_sample=1:Testing_Sample_dimension
```

```
    for tr_sample=1:Training_Sample_dimension
        temp_distance = 0;
        for i=1:Element_dimension
            temp_distance = temp_distance +
power(abs(Training_Sample(tr_sample,i)-
Testing_Sample1(te_sample,i)),2);
        end
        E_distance(te_sample,tr_sample) = sqrt(temp_distance);
    end
end

for te_sample=1:Testing_Sample_dimension
    for i=1:Training_Sample_dimension
        NN_matrix(2,i) = i;
        NN_matrix(1,i) = E_distance(te_sample,i);
    end
    NN_sort_matrix = sortrows(transpose(NN_matrix));


    for i=1:Output_dimension
        Output_matrix(1,i)=0;
    end
    for i=1:NearestNeighbor
        temp =
Training_Sample(NN_sort_matrix(i,2),Element_dimension+1);
        Output_matrix(1,temp) = Output_matrix(1,temp) + 1;
    end
    [max_value, max_index] = max(Output_matrix);
    Testing_output_matrix(1,te_sample) = max_index;
end

ConfusionMatrix = zeros(Output_dimension+1,Output_dimension+1);

for i=1:Testing_Sample_dimension
    x = Target_output_matrix(1,i);
    y = Testing_output_matrix(1,i);
    ConfusionMatrix(x,y) = ConfusionMatrix(x,y)+1;
end

for i=1:Output_dimension
    ConfusionMatrix(Output_dimension+1, i) =
(ConfusionMatrix(i,i)/sum(ConfusionMatrix(:,i)))*100;
end
for i=1:Output_dimension
    ConfusionMatrix(i,Output_dimension+1) =
(ConfusionMatrix(i,i)/sum(ConfusionMatrix(i,:)))*100;
end
temp=0;
for i=1:Output_dimension
    temp = temp + ConfusionMatrix(i,i);
end
```

```
ConfusionMatrix(Output_dimension+1,Output_dimension+1) =
(temp/Testing_Sample_dimension)*100;

ConfusionMatrix
```

APPENDIX B. Nutrition Facts

Nutrition Facts

Serving Size 1 oz. (28g/About 15 chips)

Amount Per Serving	
Calories 160	Calories from Fat 90
	% Daily Value*
Total Fat 10g	**16%**
Saturated Fat 1.5g	**8%**
Trans Fat 0g	
Cholesterol 0mg	**0%**
Sodium 170mg	**7%**
Potassium 350mg	**10%**
Total Carbohydrate 15g	**5%**
Dietary Fiber 1g	**5%**
Sugars less than 1g	
Protein 2g	

Figure A.1 Nutrition Facts – Potato chips (Lays)

Nutrition Facts

Serving Size 1 cup (28g)
Children Under 4 - ¾ cup (21g)
Servings Per Container about 18
Children under 4 - about 24

Amount Per Serving	Cheerios	with ½ cup skim milk	Cereal for Children Under 4
Calories	100	140	80
Calories from Fat	15	20	10
	% Daily Value**		
Total Fat 2g*	**3%**	**3%**	1.5g
Saturated Fat 0g	**0%**	**3%**	0g
Trans Fat 0g			0g
Polyunsaturated Fat 0.5g			0g
Monounsaturated Fat 0.5g			0g
Cholesterol 0mg	**0%**	**1%**	0mg
Sodium 160mg	**7%**	**9%**	120mg
Potassium 170mg	**5%**	**11%**	130mg
Total Carbohydrate 20g	**7%**	**9%**	15g
Dietary Fiber 3g	**11%**	**11%**	2g
Soluble Fiber 1g			0g
Sugars 1g			1g
Other Carbohydrate 17g			12g
Protein 3g			2g

Figure A. 2 Nutrition Facts – Cereal (Cheerios)

Nutrition Facts

Serving Size 1/6 medium head (89g)

Amount Per Serving	
Calories 10	Calories from Fat 0
	% Daily Value*
Total Fat 0g	0%
Saturated Fat 0g	0%
Trans Fat 0g	0%
Cholesterol 0mg	0%
Sodium 10mg	0%
Potassium 0mg	0%
Total Carbohydrate 3g	0%
Dietary Fiber 1g	4%
Sugars 2g	
Protein <1g	

Figure A.3 Nutrition Facts – Iceberg Lettuce (Dole)

Appendix C. Serving sizes used this study

Bread : 1/2 slice

Cheeseburger : one serving of cheeseburger (bread, ground beef patty and sliced cheese, 4.5 inches in diameter)

Carrot : peeled baby carrot : 3-4 pieces

Cereal : cereal with milk, 1/2 cup

Chicken : 2-3 oz

Chips : 1/2 cup of

Cookie : 1 piece (3 inches in diameter)

Fries : 3-4 rectangular pieces

Lettuce : 1/3 cup

Pizza : 1/8 piece (12 inches in diameter) (pepperoni pizza)

Beverage : 1/3 cup of soda, water and milk

(From USDA food and nutrition service)

LIST OF REFERENCES

[1] Cynthia L. Ogden, and Margaret D. Carroll, Prevalence of Overweight, Obesity, and Extreme Obesity Among Adults: United States, Trends 1960–1962 Through 2007–2008, *National center for health statistics*, 2010

[2] E. Sazanov, S. Schuckers, P. Lopez-Meyer, Non-invasive monitoring of chewing and swallowing for objective quantification of ingestive behavior, Physiol. Meas. Vol. 29, 525-541, 2008

[3] E. Sazanov, S. Schuckers, P. Lopez-Meyer, Toward objective monitoring of ingestive behavior in free-living population, *Obesity,* Vol. 17, No. 10, 1971-1975, 2009

[4] E. Sazanov, S. Schuckers, P. Lopez-Meyer, Automatic detection of swallowing events by acoustical means for applications of monitoring of ingestive behavior, *IEEE trans. Biomed eng*, Vol. 57, No. 3, 626-633, 2010

[5] E. Sazanov, S. Schuckers, The energetic of obesity: A review, Monitoring energy intake and energy expenditure in humans, IEEE Med biol magazine, 31-35, Jan/Feb 2010

[6] Amft O, Troster G. Recognition of dietary activity events using on-body sensors. *Artif. Intell. Med.*, 42:121-136, 2008.

[7] Amft, O., M. Kusserow, and G. Troster, 2009, Bite weight prediction from acoustic recognition of chewing*, IEEE Trans. Biomed. Eng*., 56, 1663–1672.

[8] Amft, O., On-body sensing solutions for automatic dietary monitoring, *IEEE CS* 2009 62-709. Amft, O., 2010 A wearable earpad sensor for chewing monitoring, *IEEE Sensors 2010 Conference,* 222–227.

[9] Nishimura, J., Kuroda, T., Eating habits monitoring using wireless wearable in-ear microphone, *3rd Int Symp Wireless Pervasive Comput, 2008. ISWPC 2008.,* 130 – 132, 2008

[10] De Belie, N., Differences in chewing sounds of dry-crisp snacks by multivariate data analysis *J sound vibration*, vol. 266, 625-643. 2003

[11] Block, G., Nutrition source in the American diet: Quantitative data from the NHANES II survey, *Am. J. Epidemiol*., Vol. 122, No. 1, 27-40, 1985

[12] Dietry Guidelines for Americans 2010, U.S. Department of Agriculture and U.S. Department of Health and Human Services, 2010

[13] Otsu, N., A threshold selection method from gray-level histograms, *IEEE Trans. Syst*., Vol. smc-9, No. 1, 62-66, 1979

[14] Joseph P. Campbell, JR "Speaker Recognition: A Tutorial". *Proc. IEEE,* 85(9):1437-1462, 1997.

[15] Makhoul J., 1975, Linear prediction: A tutorial review. *Proc. IEEE*, 63(4):561-580, April

[16] Juang B. H.; L. R. Rabiner, 1991, Hidden Markov Models for Speech Recognition, *Technometrics*, 33 (3) , 251-272.

[17] Rabiner, L. and Juang, B.H., 1993, Fundamentals of Speech Recognition. Prentice hall

[18] Tohkura, Y, 1980, Speech quality improvement in PARCOR speech analysis-synthesis systems, Ph.D thesis Tokyo Univ.

[19] P. Dhanalakshmi, S. Palanivel, V. Ramalingam "Classification of audio signals using SVM and RBFNN". *Expert Syst. Appl.* 36(3): 6069-6075, 2009.

[20] Sridharan, Sridha & Wong, Eddie "Comparison of linear prediction cepstrum coefficients and mel-frequency cepstrum coefficients for language identification" *Proc. ISIMP,* 95-98, May, 2001.

[21] Stager M., Paul Lukowicz, Niroshan Perera, Thomas von Buren, Gerhard Troster, and Thad Starner, Soundbutton: Design of a low power wearable audio classification system. *7th Int Symp Wearable Comput*, 12-17, 2003.

[22] Zaknich, A.; Attikiouzel, Y.; (1999) The classification of sheep and goat feeding phases from acoustic signals of jaw sounds. *Third Int Conf Knowledge-Based Intelligent Information Eng Syst, 1999.,* 158-161, 1999

[23] J. Weston and C. Watkins, Support Vector Machines for Multi-Class Pattern Recognition, Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

[24] Jia-Ching and Wai-He Kuok, Content-Based Audio Classification Using Support Vector Machines and Independent Component Analysis, 18th Int Conf Pattern Recognition, ICPR 2006., 157-160, 2006

[25] Lie Lu and Stan Z. Li, Content-based audio classification and segmentation by using support vector machines, Multimedia Systems 8: 482–492, 2003

[26] Chien-Chang Lin and Yukon Chang, Audio Classification and Categorization based on wavelets and support vector machine, IEEE Trans speech audio processing, 13(5), 644-651, 2005

[27] R.Shantha Selva Kumari and V. Sadasivam, Audio Signal Classification Based on Optimal Wavelet and Support Vector Machine, Int conf computational intelligence multimedia appl, 544-548, 2007

[28] J.D. Lim and B.H. Chung, The technology of the audio feature extraction for classifying contents, ETRI Technical report, 121-131, 2009

[29] O. Amft and G. Tröster, Bite weight prediction from acoustic recognition of chewing, IEEE trans biomed eng, 56(6), 2009

[30] K. Umapathy and R. K. Rao, Audio signal feature extraction and classification using local discriminant bases, IEEE trans audio, speech language proc, 15(4), 2007

[31] X. Shao and M. S. Kankanhalli, Applying neural network on the content based audio classification, ICICS-PCM, 2003

[32] J. Wang and C. Hsu, Environmental Sound Classification Using Hybrid SVM/KNN Classifier and MPEG-7 Audio Low-Level Descriptor, 2006 Int Joint Conf Neural Networks, 1731-1735, 2006

[33] Tong Zhang , C.-C. Jay Kuo, Hierarchical System for Content-based Audio Classification and Retrieval, Proc. Int. Conf. Acoustics, Speech, Signal Processing, 1998

[34] Frank Rosenblatt, Principle off Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Spartan Bookw, 1962

[35] M.L. Minsky and S.A. Papert, Perceptron, 1969

[36] D.E. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, 1986

[37] Vladimir N. Vapnik, An overview of statistical learning theory, IEEE Transactions on Neural Networks, Vol. 10, No. 5, 988-999, 1999

[38] S. Furui, Digital speech processing, synthesis and recognition: second edition, Marcel dekker, 2000

[39] T.F. Quatieri, Discrete-time speech signal processing: principles and practice, Prentice hall, 2002

[40] B. Gold and N. Morgan, Speech and audio signal processing: processing and perception of speech and music, Wiley, 2000.

[41] C. Becchetti and L.P. Ricotti, Speech recognition: Theory and C++ implementation, Wiley, 2002.

[42] Ii-seok Oh, Pattern recognition, Kyobo press, 2008.

[43] CDC Overweight and obesity, http://www.cdc.gov/obesity/