# Time Series Classification of Insect

*Yanan Jian*
Program of Very Large Information System
CMU
yjian@cs.cmu.edu

*Napat Luevisadpaibul*
Program of Very Large Information System
CMU
nluevisa@cs.cmu.edu

November 30, 2013

**Abstract**

In this project, we develop a new fast and effective way of measuring the similarity between two short sound clips. Our goal is to develop a new distance function for one nearest neighbor classifier from dataset of 1100 sound clips of 11 insect classes to predict future unseen dataset with more accuracy than any current standard distance function. We also focus on visualization part to detect any patterns among the insect sound using SVD, MDS and FastMap. Lastly we implement Local Outlier Factor(LOF) alogorithm to detect the outliers(not insect sound). We successfully develop distance function that perform better than standard distance function like Euclidean distance and cosine similarity. We also able detect 4 outliers among the several hundreds sound clips.

## 1 Introduction

*Specify the problem; Give the motivation; List your main contributions*

The problem we want to solve is the following:

- GIVEN: a collection of 1100 sound clips, of similar duration, and each having a class label among 11 classes
- FIND: a clip-to-clip similarity function
- to MINIMIZE: the classification error, in the 1-nearest-neighbor classifier.
- Visualize data from high dimensional space so that we can get intuition of properties of each classes.

- Spot outliers in dataset

This is an important problem, because standard distance function can not capture similarity between time series well; especially time series generated from a sound clip which contain a lot of noise and are in different phases.

The contributions of this project are the following:

- our proposed method is novel, combining weight distance of features extracted from frequency domain with distance in time domain and also used preprocessing steps includind wavelet transformation to extract approximation of time series, shift the starting point of time series to the same point.
- it is effective, achieving 61.55% classification accuracy far better than standard Euclidean Distance which is 35.73% accuracy
- it is scalable, being linear on the number of sound-clips $N$.

# 2 Survey

Next we list the papers that each member read, along with their summary and critique. Table 1 gives a list of common symbols we used.

| Symbol | Definition |
|---|---|
| $N$ | number of sound-clips |
| $D$ | average duration of a sound-clip |
| $k$ | number of classes |

**Table 1:** Symbols and definitions

## 2.1 Papers read by Yanan Jian

The first paper was the On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration by EAMONN KEOGH and SHRUTI KASETTY

- *Main idea*: A comparison between different time series data mining methods on real world data set.

  Sequential Scanning: Euclidian distance. Measuring similarity by calculating the distance of two points on the same time spot.

  This approach can be optimized by neglecting taking the square root. The other way to optimize this approach is to keep comparing the current best variable to the partial sums by the end of each iteration of the loop, if partial sum exceeds the current_best, we can stop iteration. Euclidian distance is a fast algorithm because it is

O(n) and sequential scan can take advantage of an optimized linear traverse of the disk.

However, Euclidian distance can not solve the problem when two time series are similar in shape but have time difference like .

Dynamic Time Warping is a technique that measures the distance between two time series after first aligning them in the time axis. It can solve the problem of two time series having the same shape but with time difference, however, the algorithm is $O(n^2)$, which can not be indexed. There is also an optimization named 'lower bounding', lower bounding is for detecting how far apart the two time series are. If the lower bound distance to a sequence current_best variable, we can prune off the fraction.

Time Series segmentation algorithms: Sliding Windows, Top Down and Bottom Up. Through the experiments from this paper, the best segmentation can bet reached by Bottom Up.

- *Use for our project*:
  We can use DTW with lower bounding method to improve efficiency when calculating distance between time series. In order to get the best segments we can use Bottom-Up algorithm.
- *Shortcomings*:
  Time warping algorithm is $O(n^2)$, besides lower bounding, we have to figure out a better way to optimize the algorithm.

The second paper was the FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets by Christos Faloutsos and King-Ip (David) Lin

- *Main idea*: Develop a fast algorithm to convert objects into points in k-dimensional space instead of using the traditional k feature-extraction functions.

  Fast Map: solve the problem of visualize high dimensional object, achieve the goal of finding N points in k-d space.. The Euclidean distances of the points have to match the distances of a given N*N distance matrix.

  1, choose pivot objects first, and project objects onto the line that passes through them in using cosine law.

  2,choose pivot: We have to ensure the line between the pivot which objects will be projected on have the advantage that the projections are as far apart from each other as possible. Traditionally, it would compute all the pairs of points which will require $O(n^2)$ computations. But the paper proposed a heuristic algorithm as: choose-distant-objects which randomly choose a point first, compute the farthest point with it, and

based on the farthest point, find another one who is the farthest from that one. Finally we can recursively find the pivots.

3, Fast Map is O(1) with respect to database size N.

4, After we reduce the dimension using fast map, we can visualize the objects in k-dimension area.

- *Use for our project*:
  1, There will be large dimensions after extracting features from objects, so we have to do a projection and draw the plots. Fast Map is linear on the database size and can reduce the N dimensions to k which we can specify.

  2, After transforming objects into points, we get large dimensions of each object, if we use Time Warping algorithm to calculate the similarity between objects, it would be extremely slow due to the unconsolidated high dimensional computing, we can try to use fast map to reduce dimension before calculating similarity.

- *Shortcomings*:
  Fast Map try to preserve as much as the distances between objects. It will at some point lose precision.

The third paper was An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback by Eamonn J. Keogh and Michael J. Pazzani

- *Main idea*: Main Idea: The traditional representation of Time Series is Fourier transformations, relational trees and envelope matching/R+ trees. But they can not perform well with noise. Piece-wise linear segmentation is a better representation of data, it connects two points with straight lines which filters noise. However, all segments are with the same weight, which is impractical in real industry. The idea is that we can add weight to linear segments during training.

  1, First we initialize the weight of all segments to one.

  2, Align the endpoints of the two sequences being compared.

  3, Merge time series, if sequence A and sequence B are in the same class (during training), we merge A and B to C, which combines information from A and B. The weight vector of C represents the the similarities between A and B. Because this is the positive training, the weights for similar segments are amplified and the differences are reduced.

  4, Negetive training, after merging A and B( from training set, different from class

A), add negative weight to B, so that the differences of the two series are amplified and the weights for similar segments are reduced.

5, Classification: Use group average agglomerative to aggregate positive examples, repeat until all positive segments are calculated and a single sequence remains. Do the same iteration on negative examples.

- *Use for our project*:
  We can use this improved Time Warping algorithm to add weight on our training set, which may generate a better result.

- *Shortcomings*:
  There's a trade off of how many segments we have to make from the time series. If there are too few segments, the weight may lead to a worse result because the representation of the time series is not accurate, then the weight will only amplify the inaccurate part. But if there are too many segments, the classification will become inefficient.

## 2.2 Papers read by Napat Luevisadpaibul

The first paper was the "Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping" by Thanawin Rakthanmanon,Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria and Eamonn Keogh

- *Main idea*: This paper proposed 4 new ways,collectively called "UCR Suite",to optimize Dynamic Time Warping(DTW) and Uclidean Distance(ED) measure. They test their algorithm with various real world dataset and compare the performance against SOTA ED(State of the Art Euclidean Distance and SOTA DTW(State of the Art Dynamic Time Wrapping)).They show that they can achieve better performance than both SOTA DTW/SOTA ED in very large data set(trillion data) making it possible to solve time series data mining problem such as motif discovery and clustering at much more larger scale. Their new four ways of optimization are as follows:

  1. Early Abandoning Z-Normalization: They incrementally compute the Z-normalization along with the ED (or LB_Keogh) of the same datapoint. In case of early abandon, they can prune both distance calculation step and normalization steps too. Note that they need normalization step because in similarity search, every subsequence needs to be normalized before it is compared to the query.

  2. Reordering Early Abandoning: They reorder the datapoint before doing the incrementally summation because different order produce different number of steps before early abandon. They conclude that the optimal ordering is to sort the indices based on

the absolute values of the Z-normalized Q becasue they assume that the query farther from mean will have larger contribution to the distance function.

3. Reversing the Query/Data Role in LB_Keogh: They calculate LB_Keogh envelope around each candidate only if all other lower bounds fail to prune instead of calculating LB_Keogh envelop around query in normal case. This removes space overhead

4. Cascading Lower Bounds: They use several of the lower bound methods in a cascade. First they use the O(1) LB_KimFL, which while a very weak lower bound prunes many objects. If a candidate is not pruned at this stage,then they compute the LB_KeoghEQ so the candidate may be abandoned anywhere between O(1) and O(n) time.

- *Use for our project*:
  We may apply these optimizations for DTW/ED to our distance function possibly the adaptation of DTW. Normal DTW computation cost is expensive thus we have to use various methods to prune as many branches as we can.
- *Shortcomings*:
  Although they claim that these methods work well with several large data set, they have not mention time series data which a lot of noises such as sound files in our insect mining project.Both ED and DTW may not work well with our data set.

The second paper was "An Enhanced Approach for LOF in Data Mining" by Vishal Bhatt. K. G. Sharma and Anant Ram

- *Main idea*: Main Idea: Generally, data in dataset follow some specific model but some exceptional data in dataset may not fit with the model. These data are called an outlier. The Local Outlier Factor (LOF) technique is efficient outliers mining algorithm that can quantify how much an outlying a certain data is in that database. The neccessary definitions to explain LOF are as follows:
  k is a user supplied natural number

  Definition 1: k-distance of an object p denoted as k- distance (p) :- the distance between object p and its kth nearest neighbor.

  Definition 2: k-distance neighborhood of an object p denoted as Nk-distance(p) :- The Nk-distance(p) contains every object whose distance from p is not greater than the k-distance.

  Definition 3: reachability distance of p with respect to object o denoted by reach-distk(p, o):- The maximum distance out of k-distance of an object o and real distance between p and o.

Definition 4: local reachability density of an object p denoted as lrd (p) :- The local reachability density of minimum points

Definition 5: local outlier factor of an object p is defined as LOF(p) = (o NMinPts(p) lrd (o)/lrd (p))/—NMinPts(p)—

The author of this paper proposed a modification in k-distance(distance between object and its kth nearest neighbor) to be m-distance(mean distance of an object and its k-distance neighborhood) and add one parameter to control the performance. This algorithm is called MLOF. They test it on two datasets and found out that number of correct detection in MLOF is greater than standard LOF with the same time complexity.

- *Use for our project*:
  We may be able to use this modified version of LOF to better detect outliers in the data. Then if we discard outliers, we may get better classifier model from training the remaining data

- *Shortcomings*:
  The author only test this algorithm on two datasets "KDDCUP99 NetworkConnectionsDataSet" and "WisconsinBreastCancerDatabase". Thus, it is not quite statistically proven that it will be doing well in other datasets too.

The third paper was "Efficient Time Series Matching by Wavelets" by Kin-pong Chan and Ada Wai-chee Fu

- *Main idea*: Main Idea: We need to reduce the number of dimension vectors to avoid dimensionality curse problem and reduce retrieval time. There are many transfomations that we can apply to our time series data such as Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Karhunen-Loeve (K- L) transform or Singular Value Decomposition (SVD). The author proposed to use Haar Wavelet Transform for time series indexing. Wavelets are basis functions used in representing data or other functions. Wavelet algorithms process data differrent way from DFT where only frequency components are considered. Wavelet has time-frequency localization property so it is able to give locations in both time and frequency. The autor choose Haar wavelet for the following reasons: (1) it allows good approximation with a subset of coefficients, (2) it can be computed quickly and easily O(n), and (3) it preserves Euclidean distance. Their methods are as follows:
  1. Pre-processing: They built index on database using an index structure such as an R-Tree by using the Haar coefficients

  2. Range Query: Similar sequences with distances less than epsilon are looked up

in the index and returned. Then find the true distances in time domain to remove all false alarms

3. Nearest Neighbor Query: They use 2 phases. First, find n nearest neighbor of query q in index. Then compute euclidean distance in full dimension between query and these neighbors. Second, perform range query using epsilon = distance of farthest neighbor in the first step. They keep a list of n nearest sequence so far and their euclidean distance. Then they keep updating the epsilon with the current farthest distance in n sequence. The nearest neighbors stored in the list are returned as answer when the range query evaluation is finished.

They do experiment on both real and synthetic data. Real data are extracted from different equities of Hong Kong stock market from 12/7/90 to 7/11/96. Their experiment showed that K-L transform gives the best precision at each dimension. On the other hand, the precision attained by Haar transform is close to the best and it outperforms DFT significantly at all except the first dimension.

- *Use for our project*:
  We can try using wavelet like Haar transformation in our project because our dataset contain a lot of noise and spike. Standard distance function such as euclidean distance or DTW may not perform very well. Also wavelet is supposed to handle spike better than Discreet Fourier Transform.

- *Shortcomings*:
  The author only test this algorithm on only one real dataset so we might want to try this algorithm on several dataset from different domain, such as our project's data, to see if the result will conform to the result in this paper.

The fourth paper was "Similarity Search Over Time-Series Data Using Wavelets" by Ivan Popivanov and Rene e J. Miller

- *Main idea*: The authors proposed methods to do similarity search over time-series data by using various wavelet transforms.

  Firt they considered whether general wavelets can be used. They showed that all wavelet with stable reconstruction (bi-orthonormal wavelets) can be used in similarity search. They mention Chan and Fu work(the third paper I read), which state that contractive property is only known for Haar wavelet.The contractive property is that the answer of our query in the feature space, when converted back to the data space, must be a superset of the original query answer so that we can ensure no false negative or false dismisal. Also they mention Fukunaga work which prove that all orthonormal transforms (Haar wavelet is one of these) preserve Euclidean Distance (thus contractive). The authors then stated further that Euclidean distance is not in general preserved for bi-orthonormal transformations. However, they proved that we can get all

the required points using sphere of radius (1/sqrt(A))E where E is query radius and A is some constant. Thus we can do similarity search using bi-orthonoramal wavelets too.

Second, they presented a result of how different wavelets perform in this application. They found that wavelets with a relatively long (12-16) filter length have the highest precision for this application. They also get this similar result on a number of real and synthetic time-series data sets too. They also show that Daubechies 12 outperformed Haar DWT, DFT and PAA(Piecewise Aggregation Approximation).

Finally, they identify classes of data that can be indexed efficiently using these wavelet transformations. The classes of data they studied are 1: the historical water levels of several lakes, 2: the stock dataset used in Chan and Fu work, 3: synthetically generated dataset.

- *Use for our project*:
  This paper have a nice performance's comparison between Haars, Daubechies, and DFT in doing similarity search over time series. One of the task in our project is to find nearest neighbor between time series data of mosquito wings flapping sound so we should try applying Daubechies 12 or 16

- *Shortcomings*:
  Although the author have done experiments on several classes of data, I don't see the dataset that contain only sound clips in their experiment. The experiment also don't acoount for noise or multiple data source in the same time series.

# 3 Proposed Method

The main motivation for our method is raw time series data are noisy and out of phase so we can not find any good distance function on data as it is. We use signal processing to extract meaningful data and classify time series based on that extracted data.

The methods we proposed are as follows:

1. Preprocess time series by Shifting all time series to have the same starting position of sound occurrence.
2. Preprocess time series by extract feature from wavelet (using the algorithm to get wavelet approximation as feature)
3. Preprocess time series by doing Fourier Transform
4. Using average peak distance as a rule for classification
5. Extract features from frequency domain for classification
6. Combine distance function from several methods above

# 4 Experiments

We implemented our method and compared it with the traditional methods.
We use leave-one-out cross-validation (LOOCV) method (using a single data from the original dataset as the test data, and the remaining data as the training data. This is repeated such that each data in the dataset is used once as the test data) to find error rate of each methods.

**Traditional Distance Function Result:**

1. Euclidean Distance : It produces Error Rate = 0.64273
2. Cosine Similarity : It produces Error Rate = 0.54636
3. Dynamic Time Warping: Error Rate = 0.63
   We improve the performance(computation time) of DTW by using UCR Suite (code provide in cpp by Eamonn in their website). We modify their code to integrate them in our system. We experimented by setting size of warping windows to be +/-5%. When we stripping the 0 from begin and end of time series, the result got slightly better (Error Rate = 0.60) but still the result is not as good as we originally expected (probably because DTW try to match bursty noise) so we decide to focus on signal transformation for now.
4. Cepstrum: It produces Error Rate = 0.43636

**Custom Distance Function Result:**

1. Shift time series to match the position of sound occurrence The below figure is a time series for insect sound in class 9

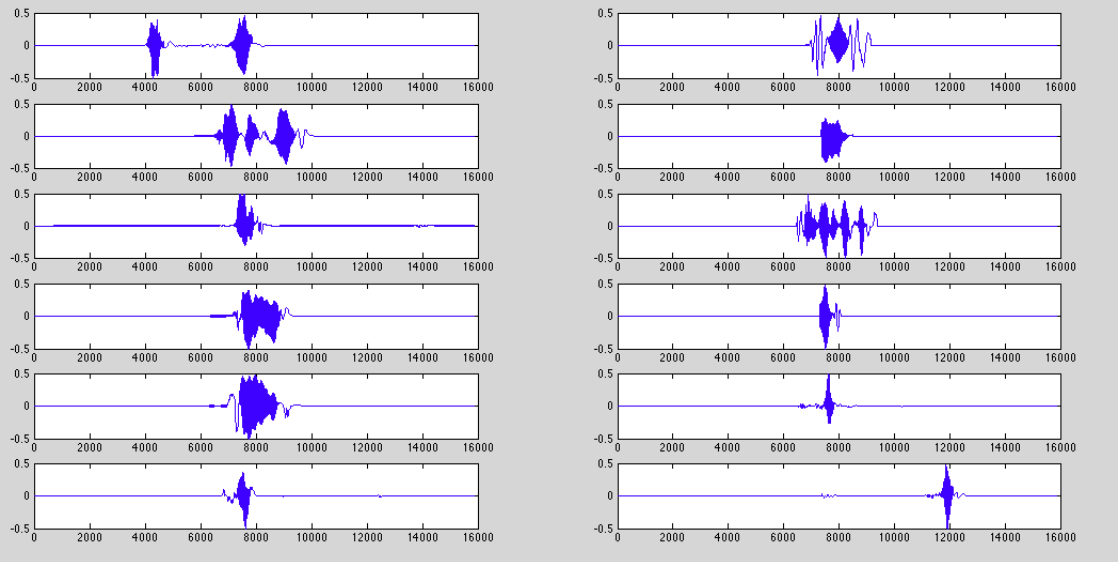


**Figure 1:** Raw data for class 9

We can clearly see that the wave in the first and the last time series are in differrent position than others. Thus these two are highly likely to be misclassified. We develop algorithm to shift the time series to begin at appoximately same location to make our distance function be more accurate. The algorithm we designed is as follow:

(a) find the first local maxima point that has height greater than 1/3 of global maxima

(b) shift the time series to the left matching position of the discovered local maxima to position 1500.

(c) keep only 4500 first time ticks because no time series has non zero elements span more than 4000 time tics.
Below is the times series in figure 1 after apply this algorithm

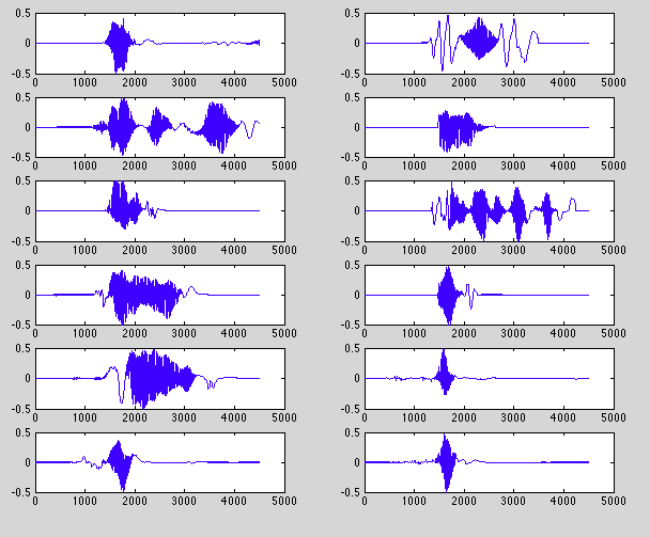


**Figure 2:** time seires for class 9 insect after shifting

The new time series above have approximately same starting position and we can also reduce the dimension by 2/3 too. Using Euclidean distance and Cosine distance on the preprocessed time series, we get this result Euclidean Distance error rate: 0.56727 Cosine Distance error rate: 0.52909

The result improves from 0.64 to 0.56 for ED and 0.54 to 0.52 for cosine)

2. Preprocessing audio signal by doing Discreet Fourier Transform.

We use Fast Fourier Transform on input signal. Then we compare the distance between transformed signal using Euclidean distance and Cosine similarity distance. The result are as follows:

(a) Euclidean Distance : It produces Error Rate = 0.40636

(b) Cosine Similarity : It produces Error Rate = 0.40091

The reason why the accurary of our classifier improves after doing Fourier Transform is because we can observe more significant trends in frequency domain rather than in time domain. For example, considering the raw time series plot of insect in class 1 below

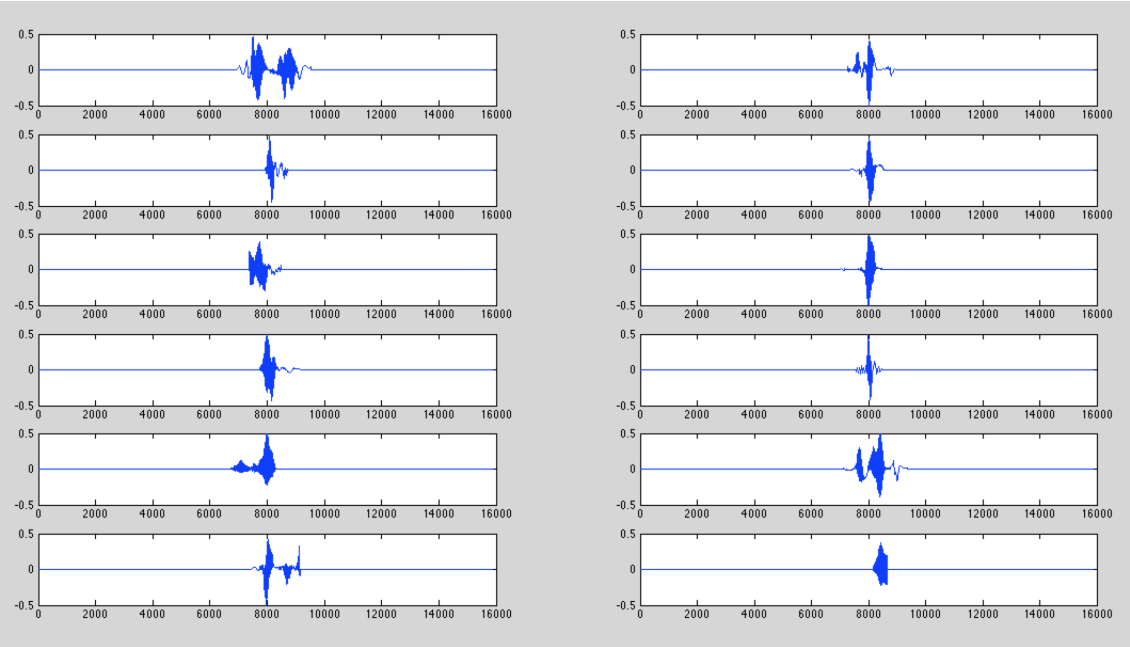


**Figure 3:** Raw data for class 1

We can not really tell what is the common pattern between these time series. But if we consider the plot in frequency domain below:

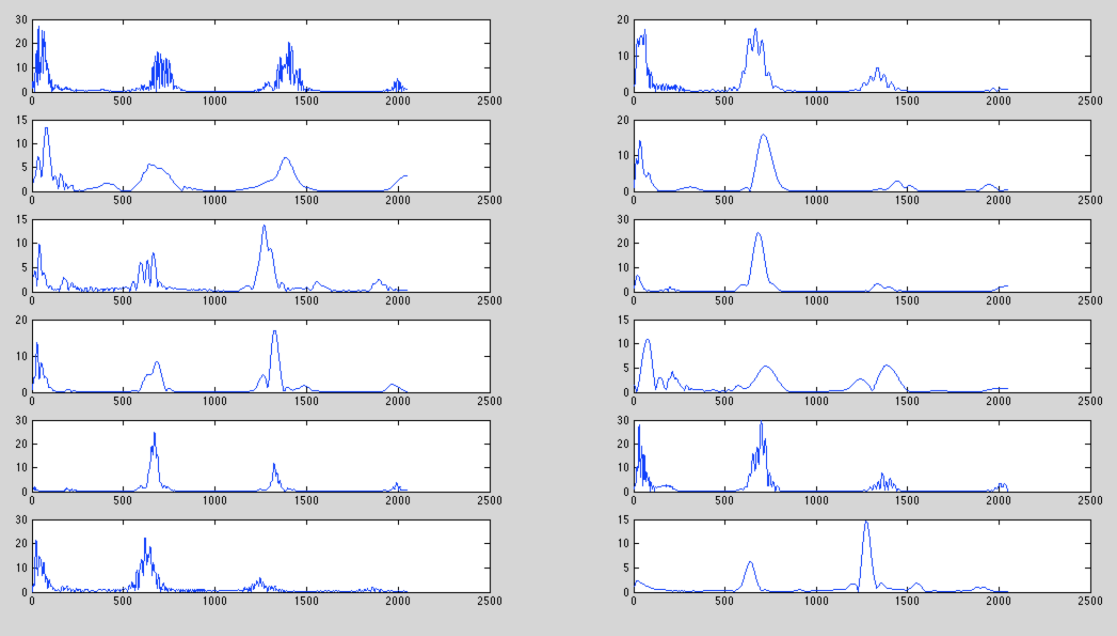


**Figure 4:** class 1 after fast fourier transform

We can observe that the position of major frequencies(first 2 or 3 peaks) is more or less similar in this class. Also Fourier Transform which doesn't capture the change in time is actually a favourable option in case that the start and end of each sounds is not in the same place. Thus apply simple distance function in frequency domain give us better result than in time domain.

Moreover we tried using SVD to extract 20 strongest concepts from our transformed signal. Then used those 20 coefficient to compare the distance using cosine similarity. We got the error rate of 0.40182

3. Preprocessing audio signal by doing Multi-Level Wavelet Decomposition.

   We tried using Daubechies-4 and Daubechies-12 wavelet decomposition at level in range 3 to 5. Then use all coefficients in every levels as features. We got the error rate at 0.54 +- 10%. This means some coefficient do not have high saperatable ability (not good for classification use). Thus our plan for next step is to find a good way to do feature extractions.

4. Preprocessing audio signal by doing Haar Wavelet Transformation, then extract feature, after that, apply Fast Fourier Transform algorithm on each time series.
   The raw timeseries of class 1 is as below: (only show 12 of them)

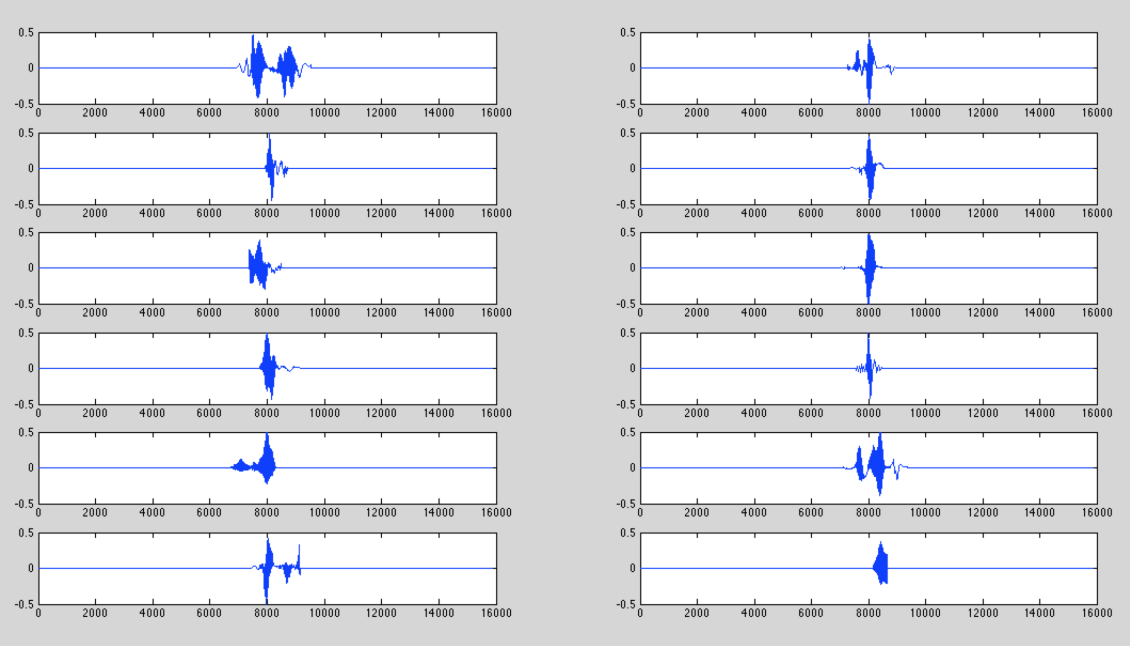


**Figure 5:** Raw data for class 1

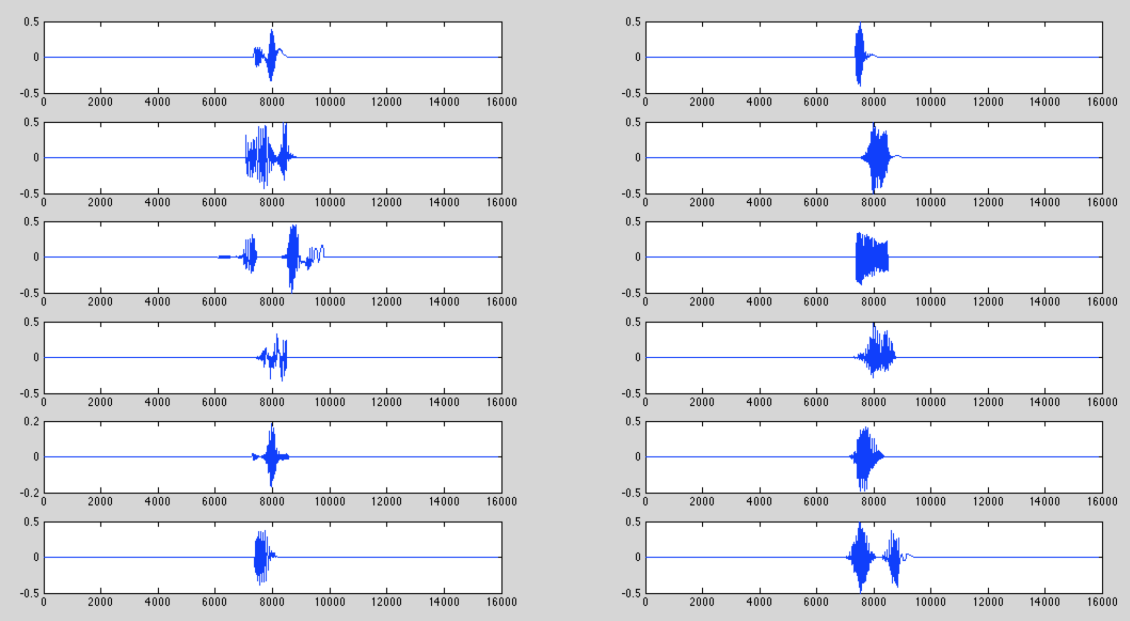


**Figure 6:** Raw data for class 2

Through observation of the above two raw time series, we can not see any pattern within one class and can not tell apart two classes of time series.
Then we apply Haar wavelet transformation and extract features from timeseries.

For Wavelet Feature Extraction,

We choose to use unsupervised classification to extract some feature. For a time series, the features corresponding to higher scale keep more wavelet coefficients and have higher dimensionality compared to lower scale. For a time series dataset having m time series, when decreasing the scale from the highest scale to scale 0, discarding the wavelet coefcients within a scale with lower energy ratio will not decrease the sum of energy of all removed wavelet coefficients to a large scale. If a scale j satisfies

$$\sum_{j=1}^{m} X_j^2 < \sum_{j=1}^{m} X_{j-1}^2$$

, removing the wavelet coefcients within this scale and higher scales achieves a local tradeoff of lower D and lower dimensionality for the dataset. If

$$\sum_{j=1}^{m} X_j^2$$

within a scale is small, there will be lots of noise embedded within this scale, cutting off the wavelet coefficients within this scale can remove more noise. The algorithm[9] is described below:

Input: a set of time-series $\{X_1^{'}, X_2^{'}, \ldots, X_m^{'}\}$
**for** $i$=1 to $m$ **do**
  calculate $\overrightarrow{A_{J-1}}$ and $\overrightarrow{D_{J-1}}$ for $\overrightarrow{X_i}$
**end for**
calculate $\sum_m E(\overrightarrow{D_1})$
exitFlag = true
**for** $j$=J-2 to 0 **do**
  **for** $i$=1 to $m$ **do**
    calculate $\overrightarrow{A_j}$ and $\overrightarrow{D_j}$ for $\overrightarrow{X_i}$
  **end for**
  calculate $\sum_m E(\overrightarrow{D_j})$
  **if** $\sum_m E(\overrightarrow{D_j}) > \sum_m E(\overrightarrow{D_{j+1}})$ **then**
    keep all the $\overrightarrow{A_{j+1}}$ as the appropriate features for each time-series
    exitFlag = false
    break
  **end if**
**end for**
**if** exitFlag **then**
  keep all the $\overrightarrow{A_0}$ as the appropriate features for each time-series
**end if**

The processed timeseries are shown below:

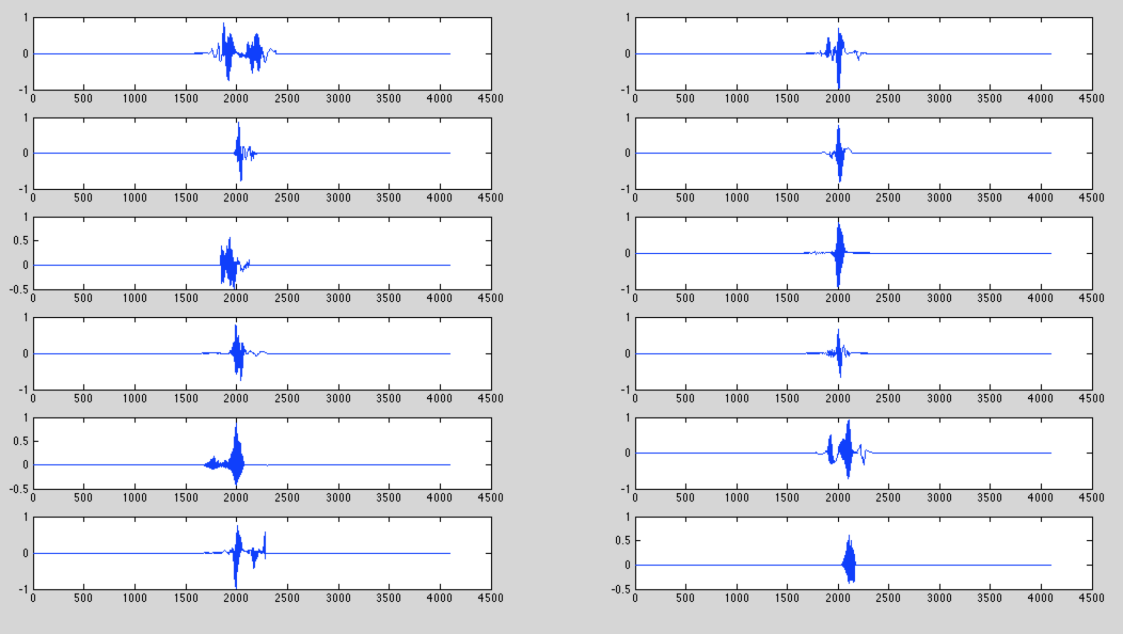


**Figure 7:** class 1 after wavelet transformation and feature extraction

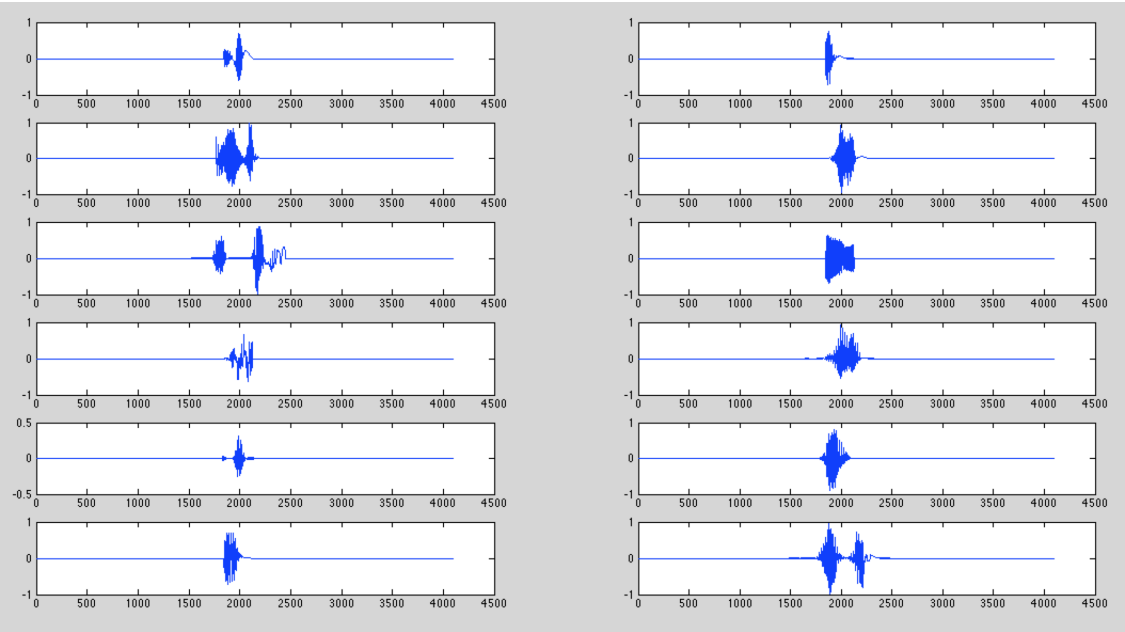


**Figure 8:** class 2 after wavelet transformation and feature extraction

We can observe that after the above steps, the dimension of timeserie is reduced and we kept the approximation as the feature of timeserie.

Now we need to find the frequency of wave from each timeseries, after apply Fast Fourier Transform Algorithm on the timeseries, we get the plots below(because FFT mirrors the plots, we choose to observe only half of each plot):

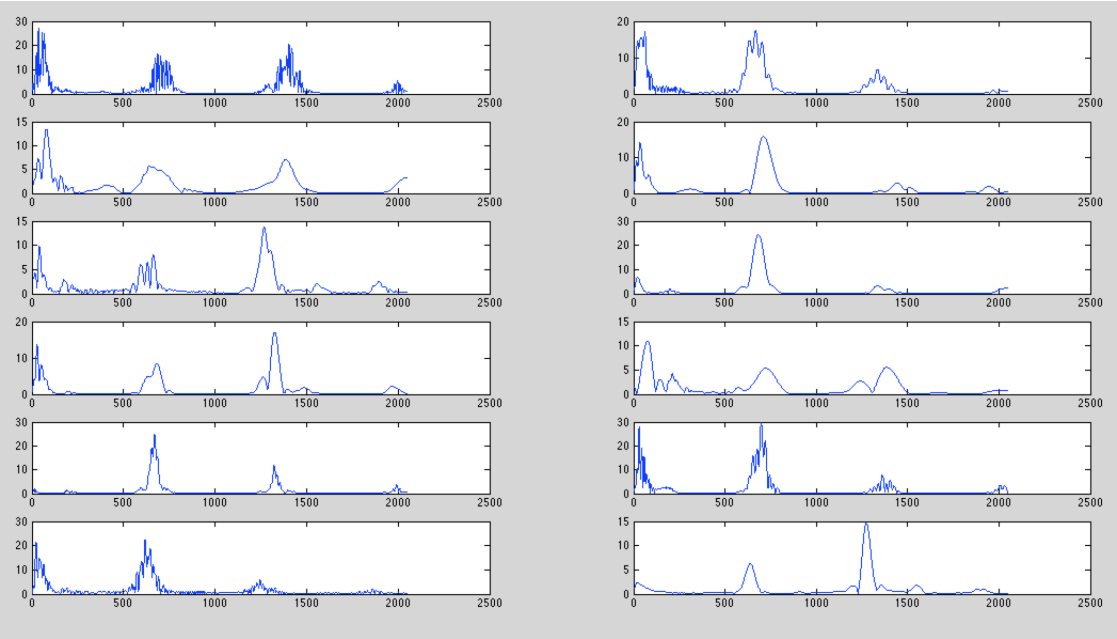


**Figure 9:** class 1 after fast fourier transform

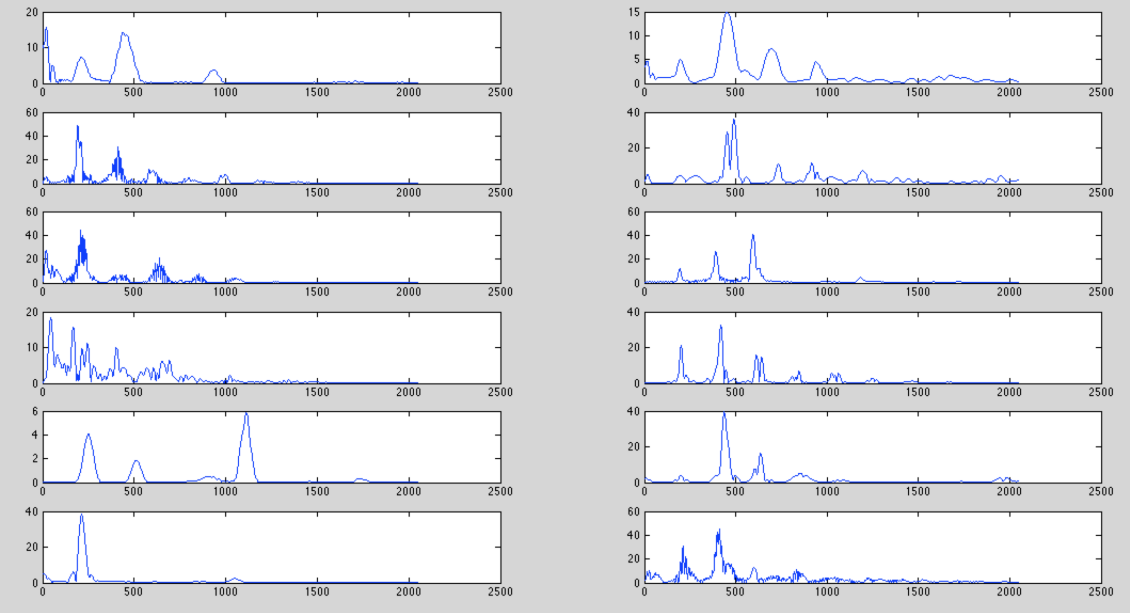


**Figure 10:** class 2 after fast fourier transform

we get the error rate of 0.40182 using Euclidean Distance function and the error rate

was 0.39455 using Cosine similarity. The result slightly improve because the feature we extract from the above algorithm is an approximation of the original time series which can also be interpreted as noise reduced version of the original wave.

5. Using average peak distance as a feature for classification
   Through observing the above figures, we found out that the distance between two neighbour peaks within one class are similar to each other. Eg. The highest peak in the first graph of Fig3 has coordinate(80, 28), the second nearest peak has coordinate(700, 18), X-coordinate represents the frequency, distance of the two peak's X-coordinate is 620, the distances between two significant peaks are roughly the same within the same class. If we observe class2, The distance is about 202, which is significantly different from the distance in class1. We suppose that we can use the difference of two frequency peaks as the feature of each timeserie, thus to do classification on these timeseries.

   The algorithm we designed is as follow:

   (a) Separate the timeseries into 80% of training data and 20% test data.

   (b) Find two significant and relatively ajacent peaks of each timeserie within the same training class and record each distance as e_dist.

   (c) Calculate average distance between two peaks found above group by class, record them as avg(dist).

   (d) Calculate standard diviation between two peaks found above group by class, record them as std(dist).

   (e) Iterate from step 2, if abs((e_dist-avg(dist))) ¿ std(dist), delete e_dist from the class group, proceed to step 3, recalculate the average distance and standard diviation of each class.

   (f) After iterating several times (we found that 2 times is enough), we finished getting rid of outliers that may influence our result, then we find two significant and relatively ajacent peaks of each timeserie within the same testing class and record each distance as t_dist.

   (g) Calculate distance between t_dist with avg(dist) of each class, classify the testing timeserie to the nearest neighbour (the class that has the shortest distance between t_dist and avg(dist)).

Result: accuracy = 41.36% (error rate = 0.5864)

Means of distances in Training Set is shown in Table 2:

**Table 2:** Mean Distances

| Class Label | Mean Distance |
|---|---|
| 1 | 613.1786 |
| 2 | 198.5263 |
| 3 | 354.4655 |
| 4 | 306.5439 |
| 5 | 509.0625 |
| 6 | 345.2319 |
| 7 | 499.6552 |
| 8 | 374.5667 |
| 9 | 521.0877 |
| 10 | 219.4545 |
| 11 | 166.9000 |

Observing the mean distance of each class above, the reason that the accuracy is low may due to the unobvious differences between the distances, Eg. class 5, class 7 and class 9 have similar distance values, thus it may result in misclassify when classifying testset.
Plot class 5, class 7, class 9 on Fig 7,8,9 as below:

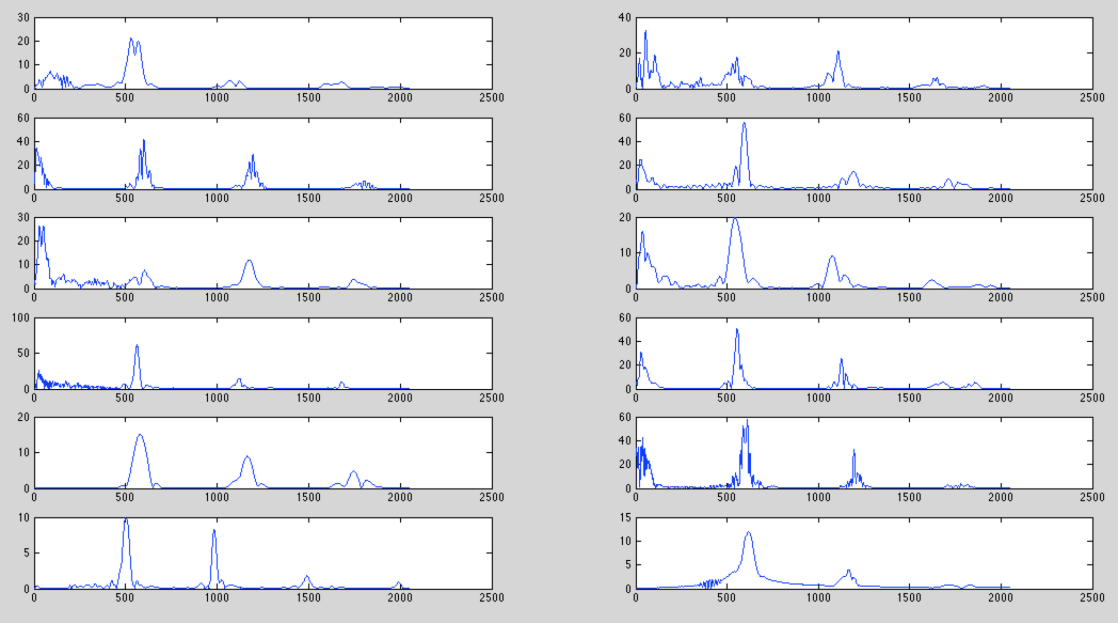


**Figure 11:** class 5 after fast fourier transform

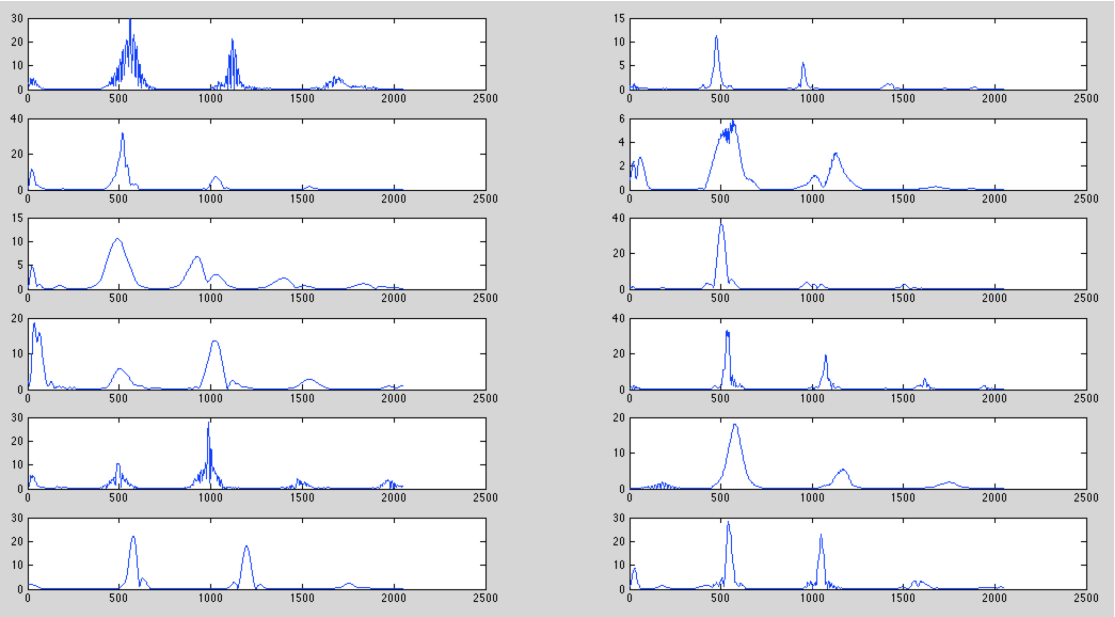


**Figure 12:** class 7 after fast fourier transform

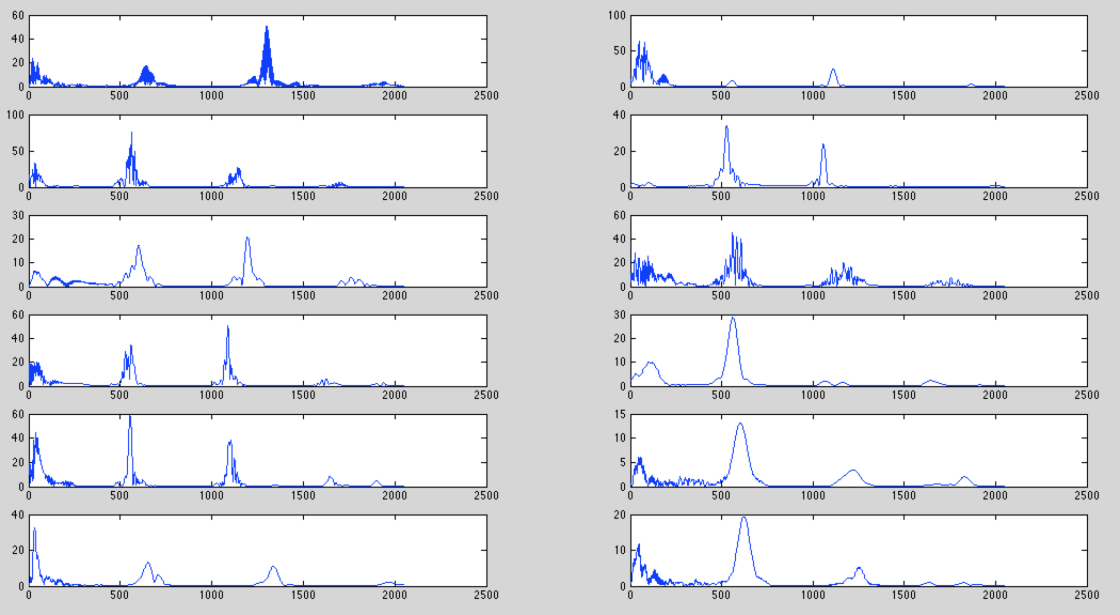


**Figure 13:** class 9 after fast fourier transform

From the plots above, we can see that class 5,7,9 have similar distances between the two most significant peaks.

6. Use features extracted from frequency domain for classification
   In the previous method we've found that average distance between peak is not informative enough to classify this dataset. Thus instead of just using average distance between peak, we did experiment on various features extracted from data as follows:
   Note: we now use LOOCV to do all the remaining experiments and we use standardized Euclidean distance because each feature is in different scale so we have to normalize it first.

   (a) Use highest frequency and second highest frequency as two features (this correspond to position in X axis that have highest and second highest peak in Fourier plot). The intuition is that similar insect sound should have nearly the same major frequency. We do not use 3 peaks because many time series have no more than 2 peaks that pass minimum peak height threshold. The error rate is 0.51091 which is better than using average peak distance

   (b) Add three more features: distance between highest and second highest peak, amplitude of highest peak and amplitude of second highest peak. We hypothesized that insect sound in the same class should share these charactoristic too. The result error rate for using all five features is 0.48364, sligly better than using two

features

(c) Add number of peak found as another feature. We found that give more weight to this feature ie use square of number of peak give slightly better result. Number of peak found for each wave usually range in 2 to 4. We squares it to make distane further apart. The error rate is 0.46455

(d) We now have six features for distance function and we want to find out whether we can get rid of some features so we did experiment by removing one of features. We've found that no combination of five features beat performance of six features

(e) We experimented using one feature at time and get this result

**Table 3:** Error rate for using single feature

| Feature | Error Rate |
|---|---|
| Position of the highest peak | 0.61 |
| Position of the second highest peak | 0.71 |
| Distance between two peaks | 0.6 |
| Amplitude of the highest peak | 0.89 |
| Amplitude of the second highest peak | 0.91 |
| Number of peaks | 0.91 |

Thus we conclude that each feature does not have much discriminative power by itself but we can combine several features to get more information on the data and improve classification accuracy.

7. Custom distance functions by combining different schemes
From all of the above experiments. The best result is come from cosine similarity on Fourier transform. We want to improve this by combine this with distance function get from the previous experiment. The methods we tried are as follows:

(a) We combine distance of standardized Euclidean on Fourier Transform wave and tandardized Euclidean on six features extracted from the wave. The error rate was 0.40

(b) We use approximation at level 1 using wavelet transformation to reduce the noise before calculating distance function(FFT, extract features, calculate distance). This came from and observation that if we follow the algorithms proposed by Hui Zhang et al, the approximation at level 1 is enough for this data set. The result of using different mother wavelet are as follows:

**Table 4:** Error rate from using distance function after wavelet transformation

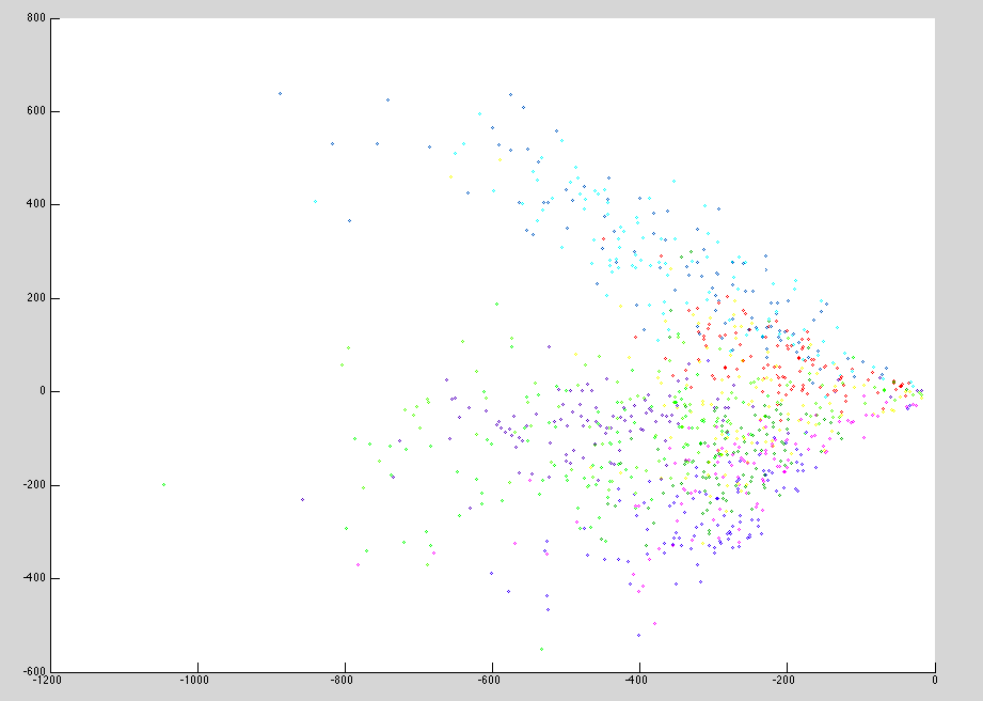
| Wavelet used | Error Rate |
|---|---|
| haar | 0.40273 |
| db4 | 0.38909 |
| db8 | 0.40273 |
| db12 | 0.38727 |
| db16 | 0.39364 |
| coif1 | 0.40273 |
| coif3 | 0.39273 |
| coif5 | 0.39091 |

8. Adding standardized euclidean distance function on time series in time domain (with shifting to match position of the sound).
   We add this as a bias in case that some time series are very differrent in time domain but happen to be similar in frequency domain. In first attempt we just add three custom distance function together with the same weight ie distance using features + distance in frequency domain + distance in time domain. The error rate was 0.44727. The result get worse because distance in time domain does not perform as well as the other two distance functions, so we put more weight to distance in frequency domain and features using this formula: cube(distance using features) + cube(distance in frequency domain) + distance in time domain.
   The error rate was 0.38455 which is the best we get from all experiments.

**Visualization**

(a) SVD
   We use SVD to reduce the dimensional of transformed signal to two dimension by keeping only two strongest concepts. We assign different colors to each insect classes and plot the result using scatter plot.

Experiment result:




We can detect some patterns from the plot. For example,the red class clustered at the central right part of the graph, the density of the red class looks higher than any other classes. The dark blue class scattered within the right part of the graph, the light blue class seems linearly distributed. The high density cluster in the right center part area is a mixture of many classes so it's hard to classify.

(b) MDS
Using MDS with custom distance function in frequency domain we got below result:

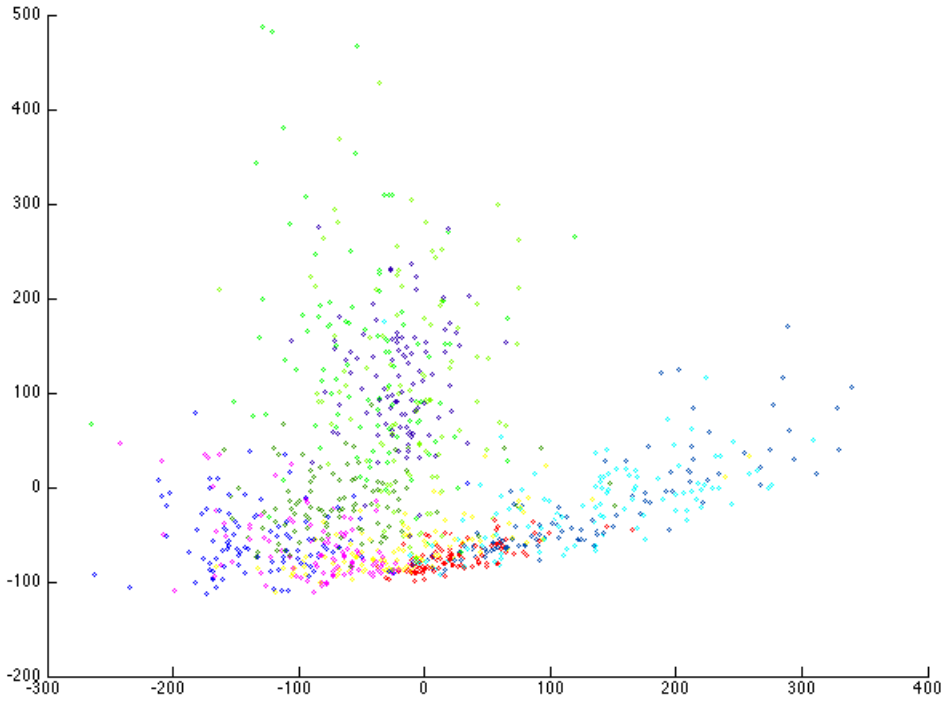



**Figure 14:** Visualization Using MDS

We can detect several clusters from this plot. The light blue cluster is concentrate on the right side and look linear. Light green is on the upper side. On the left we spot pink cluster and a group of dark blue. Red and yellow concentrate on the bottom part.

(c) FastMap
Using Fastmap with custom distance function in frequency domain we got below result:

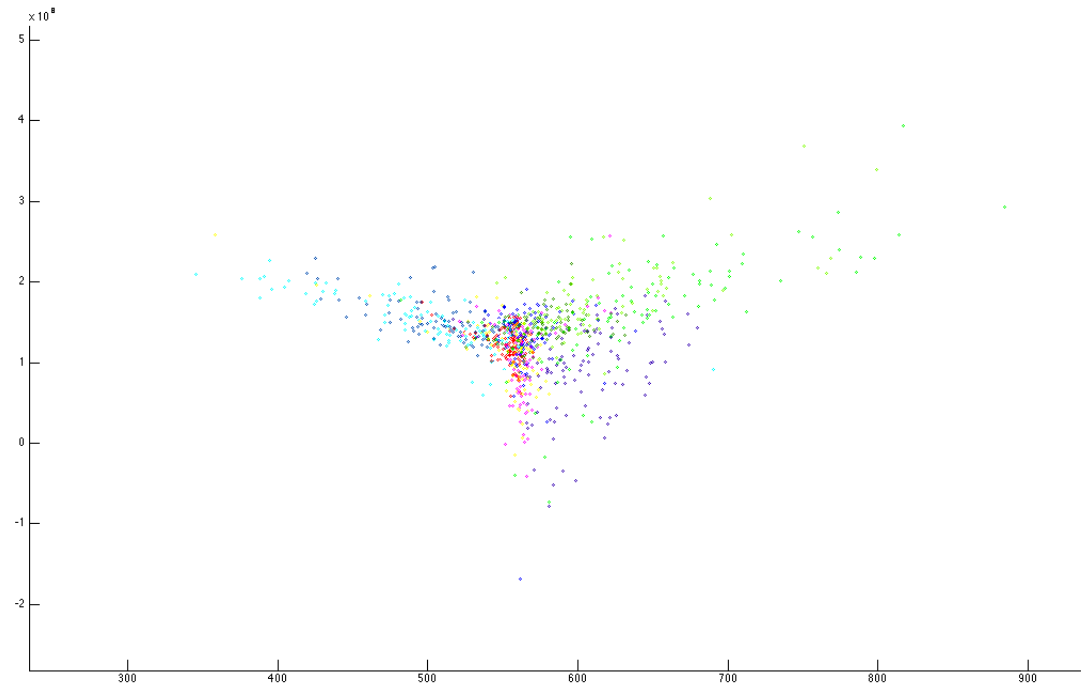



**Figure 15:** Visualization Using FastMap

We can spot more or less similar pattern in FastMap and MDS. Green cluster is now on the right side; while light blue is on the left. Pink cluster is on the down side near the center and most of the cluster concentrate on the center. The drak blue seem to be scatter around. The dense area in the center is hard for classification.

**Anomaly Detection**

We used LOF as the algorithm of detecting anomalies. The algorithm indicates that the detection of anomalies can depend on how isolated the object is with respect to the surrounding neighborhood. Through calculating each object's Euclidean distance with other objects and set the number of k(k-neighbours) from 10 to 100, after comparing the distances, we found four wav files' LOF value always ranked top four, so we believe these four are anomalies:
00066.wav
00017.wav
00128.wav
00007.wav
Their LOF value to other objects are significantly larger. The distances are:
18.7306
16.3915

14.1451
2.6638
while others all have the LOF value less than 2. The outliers found above are plotted as follows:

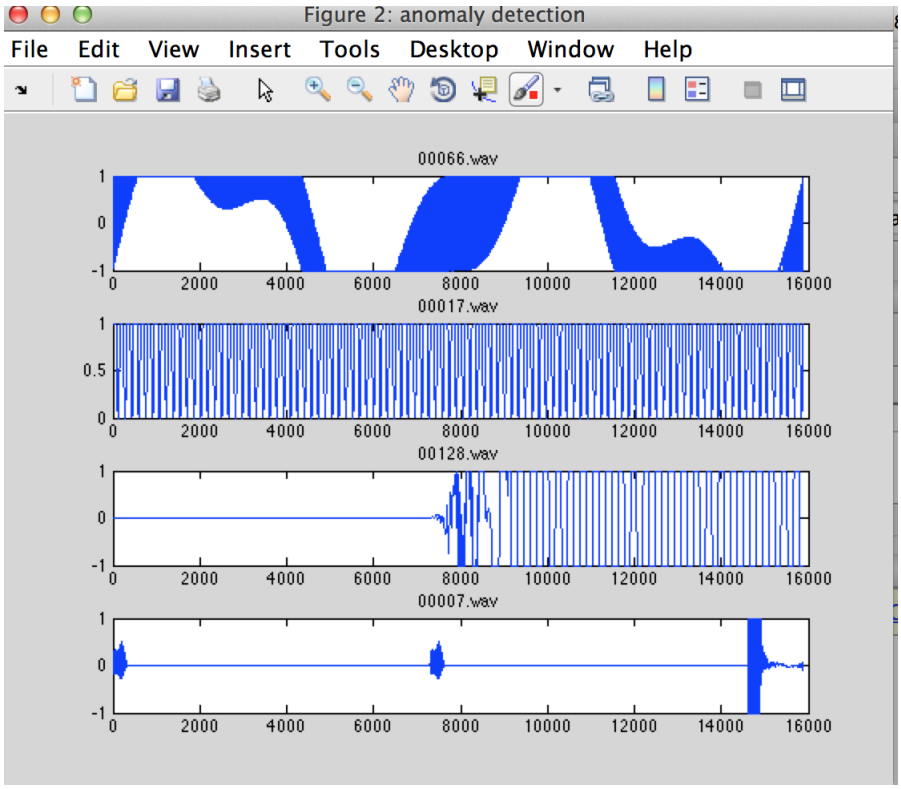



**Figure 16:** Outliers

Compare with the normal ones shown below:

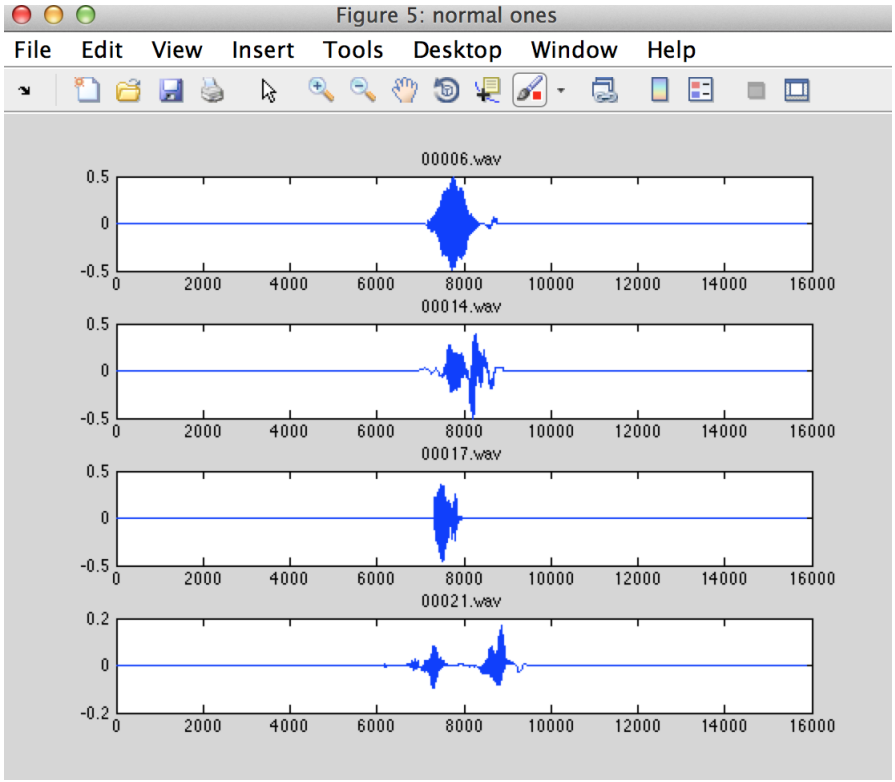



**Figure 17:** Normal ones

We believe that we correctly selected the anomalies.

# 5 Conclusions

The proposed method to use features in frequency domain and combine with distance in time domain has the following advantages:

- it gives better classification accuracy than all of the methods we have tried
- its accuracy is better than all standard distance function
- it is scalable as it scale linearly with data.

The below table show the Error rate (1-accuracy) of all methods we've tried

**Table 5:** Error rate of each methods

| Method | Error Rate |
|---|---|
| Euclidean Distance | 0.64273 |
| Cosine Similarity | 0.54636 |
| Dynamic Time Warping | 0.63 |
| Cepstrum | 0.43636 |
| Shifting + Cosine | 0.52909 |
| Fourier + Cosine | 0.40091 |
| Wavelet + Fourier + Cosine | 0.39455 |
| Average peak distance | 0.5864 |
| Fourier + Feature extraction | 0.46455 |
| Wavelet + Fourier + Feature extraction | 0.38727 |
| Custom distance function using combined method | 0.38455 |

# References


(a) T.Rakthanmanon,B.Campana,A.Mueen,G.Batista,B.Westover,Q.Zhu, J.Zakaria and E.Keogh. 2012. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. KDD.

(b) V.Bhatt,K.G.Sharma and A.Ram. 2013. An Enhanced Approach for LOF in Data Mining. International Conference on Green High Performance Computing.

(c) K. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In Proc. of the ICDE Conf., pages 126133, Sydney, Austrialia, 1999.

(d) I.Popivanov and R.J.Miller. 2002. Similarity Search Over Time-Series Data Using Wavelets. Proceedings of the 18th International Conference on Data Engineering.

(e) Eamonn Keogh , Shruti Kasetty. 2002. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. SIGKDD'02.

(f) Christos Faloutsos , King-Ip (David) Lin. 1995. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets.

(g) Eamonn J. Keogh and Michael J. Pazzani. 1998. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback.

(h) Hui Zhang and Tu Bao Ho School of Knowledge Science, Japan Advanced Institute of Science and Technology, Unsupervised Feature Extraction for Time Series Clustering Using Orthogonal Wavelet Transform

# A Appendix

## A.1 Labor Division

The team performed the following tasks

- Implementation of K-Nearest Neighbor classification [Napat]
- Implementation of Euclidean distance and Cosine similarity [Napat]
- Implementation of Dynamic Time Warping [Yanan]
- Integrate UCR Suite for DTW [Napat]
- Experiment on DTW(w/o 0s stripping)[Napat]
- Experiment on Cepstrum [Napat]
- Preprocessing signal using Fourier Transform and Wavelet decomposition [Napat]
- Doing dimensional reduction using SVD [Yanan]
- Extract concepts from SVD and plot the result [Napat]
- Analyse scatter plot [Yanan]
- Preprocessing signal by shifting time series [Napat]
- Feature extraction for wavelet coefficient [Yanan]
- Experiment on average peak distance method [Yanan]
- Experiment on features extracted from peak information [Napat]
- Experiment on combining distance function [Napat]
- Visualization using MDS [Napat]
- Visualization using FastMap [Yanan]
- Implement LOF algorithm and find outliers [Yanan]

# Contents