

KEYBOARD CLASSIFICATION



Abstract of  
a thesis presented to the Faculty  
of the University at Albany, State University of New York  
in partial fulfillment of the requirements  
for the degree of  
Master of Arts  
College of Arts & Sciences  
Department of Mathematics & Statistics

Matthew D. Lester

2010

## ABSTRACT

A lot of sensitive information is transmitted in the social world. Although the visual components are usually shielded from others, the auditory components are almost never disguised. This is because the human ear isn't able to decipher these signals well. This thesis used Statistical and Machine Learning methods to discriminate between computer key strokes using only sound. Specifically, they included: Support Vector Machines (SVMs), Mel Frequency Cepstral Coefficients (MFCCs), autocorrelation, binary classification, and linear regression. The programming of these techniques was done for the R statistical package.

The methods used to solve the classification problem between two keys was successful. The algorithm required modification to maintain an acceptable level of accuracy with respect to random chance for the four key classification problem.

Further investigation would be required to truly apply this in the world; however, the results that were obtained suggest that it is possible.

# KEYBOARD CLASSIFICATION

A thesis presented to the Faculty  
of the University at Albany, State University of New York  
in partial fulfillment of the requirements  
for the degree of

Master of Arts  
College of Arts & Sciences  
Department of Mathematics & Statistics

Matthew D. Lester

2010

## ACKNOWLEDGEMENTS

It is a pleasure to thank those that made this thesis possible.

I owe the deepest gratitude to my thesis advisor, Carlos Rodriguez. Carlos expanded the way I think about mathematics in the most wonderful manner. The enthusiasm and dedication I have towards the mathematical field of statistics today is due to the continued support, and insightful teachings of Professor Rodriguez. Thank you for bringing your passion into the classroom and for motivating me to reach for the highest aspirations.

Working with my graduate advisor, Karin Reinhold over the course of my academic career was an amazing experience. Karin played a large role in the decision I made to go into the field of Statistics, and I will never be able to thank her enough for the guidance that she has given me.

It was with great enjoyment working with Martin Hildebrand. This thesis would not have been able to be completed without the work that we did together. I would like to thank you for giving me the opportunity to work with you, and for the constant feedback which helped me gain confidence in the field of mathematics.

I would especially like to thank my good friend, and colleague Nicholas Jarrett. Writing our theses was a testament to the passion that we both share for mathematics. I will not forget the endless nights that we spent completing our research.


Lastly, I would like to thank my family for continuing to support me with every decision I make.

## TABLE OF CONTENTS

ABSTRACT.....	ii
ACKNOWLEDGEMENTS.....	iv
TABLE OF CONTENTS.....	v
CHAPTER	
1 PRIOR INFORMATION.....	1
2 ISOLATING SIGNALS.....	5
3 MFCCS & AUTOCORRELATION.....	8
4 SUPPORT VECTOR MACHINES.....	11
5 THE TWO KEY CLASSIFICATION PROBLEM.....	15
6 THE FOUR KEY CLASSIFICATION PROBLEM.....	21
7 CONCLUSION.....	26
REFERENCES.....	27

# 1 Prior Information



When a computer keyboard key is hit, a l is produced. This signal is unique to the fact that it is indistinguishable to the human ear. The two key classification problem is concerned with discriminating two key strokes using labels and the sound generated from the key stroke. This problem was explored in [1]. In this thesis, different techniques are used to try and gain the same results or better.

The signals that are produced when hitting a key on the keyboard are representations of its spatial and physical characteristics. This problem samples the signals at specific points in time, and thus is considered to be a discrete signal processing algorithm. To record all of the samples of keys on the keyboard, the program Audacity was used. A picture of the wave associated to the A-key for 50 key strokes can be seen in [Figure 1].

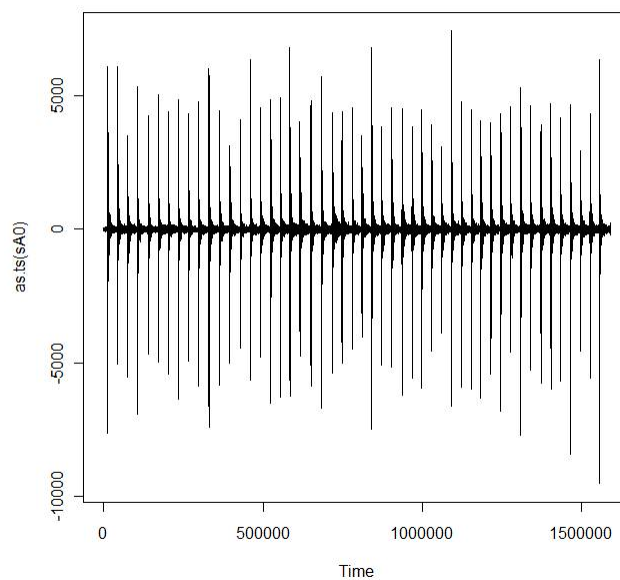


Figure 1: Wave Associated to the A-Key for 50 Hits

When sampling signals there is normally a filter that is primed to reduce the data space. This is necessary because the raw signals are too large to work with in an analysis. The filter used on the signals is called the Short Time Fourier Transformation (STFT). The signals are transformed from the time domain into the frequency domain using the STFT. By transforming the signal into the frequency domain, the signal is being prepared for an easier analysis for the two key classification problem, and later the four key classification problem. The frequency domain will give a representation of the amount of a signal incorporated into frequency bands over a range of frequencies. This domain will allow for features to be extracted from the signal.

To fully understand the problem of key classification it is desired to use the most important prior information to create a worthwhile analysis. In the case of key classification, the prior information consists of the physics of the keyboard and the user. After obtaining this prior information, it will be incorporated into functions of the analysis. This will be discussed further in the Isolating Signals section, which focuses on key hit extraction techniques.

The physics of the keystrokes is an important part of this problem. This had to be studied carefully to obtain the most reliable prior information before beginning analysis. A human can type on average up to 300 keys per minute on a standard keyboard [2]. When a human hits a single keystroke there is a push and release of the physical key. This push and release mechanic is what generates the signal. From [2] there are approximately 100 ms left between subsequent keystrokes. This fact was used as prior information when forming windows around the isolated key hits. This prior information is considered to be of human nature, where the person who is typing will attribute unique characteristics to the string of characters typed.

A key on a computer keyboard consists of three components. A figure of a key taken apart can be seen in [Figure 2]. The three components are the head, a dome-shaped rubber part, and a plastic connector between the head and the dome piece. Individual keys are held together by a metal base plate [2]. Each key has an electrical switch embedded, when a key is pressed the rubber part is squeezed and the top of the dome forces pressure onto the electrical switch.

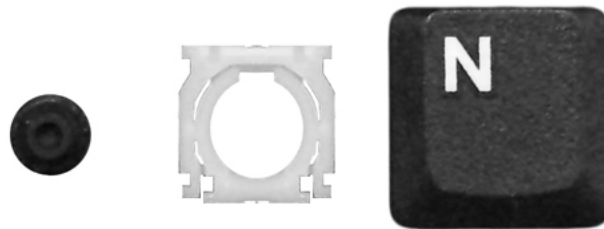


Figure 2: Key Components

Based on [2], it was hypothesized that keys may produce different sounds according to three factors:

1. There are slight differences in the construction of the 30-character keys. This naturally leads to differences in the signals they respectively produce. Certain keys are clearly more similar than others. For example, the space key has a unique signal that can be easily distinguished from other characters even with the naked ear. Likewise, keys with more similarities in their physical properties, are harder to distinguish between.
2. The signal produced by a key may include traces of the signatures associated with neighboring keys.. The idea is essentially that nearby keys resonate with a keystroke, thus adding a small amount of their own unique signature to the total wave.

3. The keyboard plate may act as a drum, which would resonate outward with each press of the key at different spatial locations.

The second conjecture was proven false in [2], while the third factor would explicate the reason for individual keys producing unique signals. The last hypothesis was proven true by removing the metal plate, and cutting out several pieces associated with specific keys. The remaining keys were placed back on the keyboard and analyzed. The techniques used in the analysis were unable to distinguish keys after this procedure because the relative position of the keys being classified was unrecognizable.

This leads to spatial prior information that can be extracted from the keyboard. As stated before, the computer keyboard has a metal plate in which the keys are connected to. This plate operates like a drum, and has a unique resonance signal based on the spatial location of the key being stroked. In other words, keys that are on the edge of the keyboard will have a different resonance than keys that are in the middle of the keyboard. From the prior information it was shown that any two keys can be discriminated and classified based on the signal from the keystroke. This was proven in [1].

## 2 Isolating Signals

The R code for the following analysis is located at [7]. To be able to analyze the key signals, the individual key hits must be isolated in the sound file. The function GetHits (GH) was developed to complete this task. GH uses the prior information of physics along with standard regression techniques to give an accurate portrayal of the isolated signals. It then scans through the signal produced by the sequence of key hits and gathers the top peaks subject to a minimum separation condition, until the function reaches the maximum keys that it is required to search for. The maximum keys are set to 50 in the problem posed, but can be changed to as many keys the user types and intends for GH to search for.

The minimum separation is set to 30 units, which is a measure of closeness of peaks. This ensures that there are no missed key hits, and that the function is not sampling multiple values from the same key-hit. The minimum separation was chosen according to the prior information of physics. Based on the physics of the keyboard a minimum separation between key hits exists, thus justifying the existence of the minimum separation condition. For example: GH is going to pick a maximum of 50 top peaks, and observes a very loud hit with a large maximum peak. By prior information there will also be larger values observed close to, and around the observed hit. Therefore, some of the 50 top peaks chosen may be from the same key-hit. GH will take care of this problem and throw out all of the top peaks that fail the minimum separation condition. If the number of top peaks remaining is less than the maximum keys declared, GH increments the number of top peaks it searches for by one. This process is then repeated until all the conditions are satisfied.

After finding the maximum peaks according to these conditions, it was important to find out when a key-hit starts and furthermore how long the key-hit lasts. To get

an idea of how long a key-hit lasts, it is necessary to realize the physics of an actual keystroke. The time of the maximum observed peak corresponds with complete compression of the key onto the metal base plate. Following this, is the release of the key which is characterized by a smaller peak. The combined signal decays exponentially fast until reaching a state of reduced variability. By studying the isolated signals in [Figure 3], a reasonable approximation for the length of the key-hit was found to be 350 units. The push and release of the key were easily viewable by eye. Prior to the push of the key the signal would produce spikes [Figure 4], and posterior to the release of the key the signal would produce hills and valleys [Figure 5]. In the GH program there is a consistent end cut-off point used for each isolated key signal. This end cut-off point was chosen subject to minimizing the amount of information lost while isolating the signal, and making sure that the hit of the key was actually in the window that was chosen.

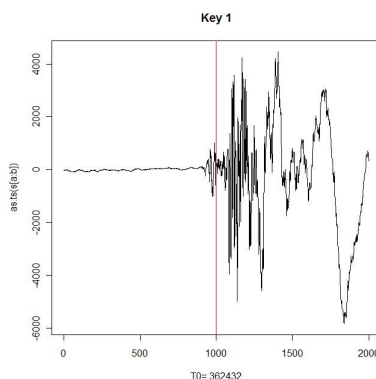


Figure 3: Isolated A-Key Signal

It seems logical to assume that the majority of information is encoded in the actual hit, or in other words when the physical key is fully compressed onto the metal plate. In reality, there is going to be a certain amount of information lost in the tail after the cut-off point. In a future analysis it would be of interest to see how much information is held in the tail of the isolated signals in pursuit of approximating an optimal window. These facts are assumed for the thesis and worked well when

isolating the signals.

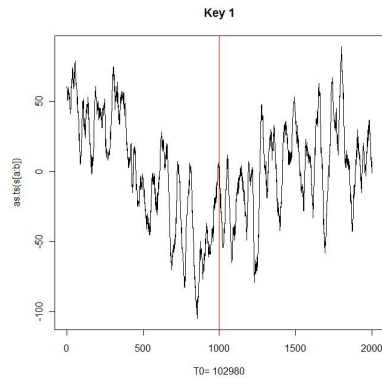


Figure 4: Prior to a Hit

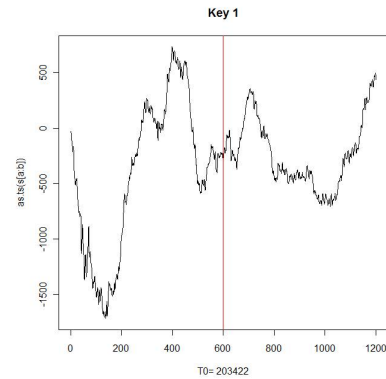


Figure 5: Posterior to a Hit

To get the correct start time of a hit within the isolated signal many sophisticated methods were tried, but none behaved as well and as predictably as linear regression. The functions used to identify the start of the hit are called LReg and Thresh with tuning parameters tau and lambda respectively. The tuning parameter in Thresh,  $\lambda$  is used as a minimum height requirement.  $\lambda$  is the % of the overall maximum amplitude achieved within the signal or maximum top peak. The value 0.085 was experimentally found to be a good estimate for  $\lambda$  in GH. Therefore, any signal in the bottom 8.5 % of amplitudes is rejected. This minimum height signifies the first compression of the isolated signal, or the detection of the start of a keystroke. Thresh is used as a subfunction within LReg. These functions distinguish between keystrokes and silence using the energy dispersed within the isolated signal windows. The tuning parameter  $\tau$  is used in the piecewise regression computed in LReg, and specifies the number of standard deviations used to measure the variability within the region of interest. This regression is used to measure the variability between regions and assists in the identification of the minimum threshold.



### 3 MFCCs & Autocorrelation

Even though the signal has been greatly reduced, it's still very difficult to do analysis on the remaining pieces. The isolated signals contain a large amount of information, and for the purposes of this thesis there was not enough computing power to use the raw isolated signals for analysis. Therefore, techniques were imposed onto the isolated signal to further reduce the data size subject to the condition of maintaining the most important characteristics.

MFCCs or Mel-Frequency Cepstral Coefficients was the best suggested method to use according to [1] and [2] for the most prominent feature extraction. MFCCs are typically used for speech recognition problems, but was shown to be successful for discrimination of keyboard keys in [1] and [2]. The MFCCs use the Mel-Frequency scale which has an intensity function which is intended to model the human auditory system. After the feature extraction using MFCCs, each isolated signal of a keystroke is now represented as a vector of features. This vector of features or MFCCs is a greatly reduced representation of the isolated signal.

More explicitly, the MFCCs are extracted from the signal using the following procedure:

1. Convert the signal into the frequency-domain using the Short-time Fourier Transforms (STFT). STFTs are Fourier Transforms (FT) over small incremental windows which span the entire signal. They are of the form  $e^{-j\omega t}$ . These results are stored within a matrix denoted STFT.
2. Filter the results using the Mel-scale intensity function and the desired number of coefficients. Store the results into a scaling matrix. The filtering for the scaling matrix divides the frequency domain into overlapping chunks, which are

called critical bands.

3. Sum the frequencies in each band by composing the scaling matrix with the STFT matrix, or  $[STFT] \times [Scaling]$ .
4. Take the logarithm of each sum, and negate the result.
5. Compute the Inverse Discrete Cosine Transform (IDCT) of the logarithms. The IDCT can be interpreted as an inverse FT. This step essentially undoes the transformation of the signal into frequencies using FT from 1.
6. Compose the results from 3. with 5. to obtain the extracted frequencies of the signal. This final equation is of the form:  $-\log ([STFT] \times [Scaling]) \times [IDCT]$

Another method of feature extraction that was used along with MFCCs was the autocorrelation function. Using lags ranging from 2 to 66, resulted in 65 coefficients being produced per isolated signal. These coefficients, when added with the MFCCs, increased the accuracy and predictability of the SVM. The autocorrelation function measures instantaneous changes in the model, and therefore random pieces in key A are different than random pieces in key L. In general, the correlation between terms with a small lag between them is very high. In other words, when there is a lag of zero the maximum correlation value will occur since the signal will always be perfectly correlated with an exact copy of itself. However, when the lag is increased, the effects that occur between the terms can reduce the correlation. This will occur if the autocorrelation function identifies different patterns in the isolated signal. The way in which these correlations decrease is characteristic for each key we tested and allowed us to increase our predictive power.

The autocorrelation of a signal will reveal patterns within the series of values that are being analyzed. It is of interest to analyze the series of values in an isolated

key-hit, to extract a pattern using the autocorrelation function. Furthermore, the autocorrelation function will detect variability within this pattern. For example, the model may be steady for a period of time with low variability, and then rapidly have a change in variability which translates to something is happening at this exact point in time when the model instantaneously changes. In this specific example the correlation would decrease. The small windows around the occurrence of these changes are used to create the autocorrelation coefficients. These coefficients will be used as features for the isolated signal being analyzed. It is important to note that the unique characteristics for the A-key will be different than those of the L-key or any other character. Therefore, the autocorrelation coefficients when added with the MFCCs, will drastically reduce the data set while still being able to recover important characteristics of the original raw signal.

## 4 Support Vector Machines

To be able to talk about support vector machines (SVM) and how they work, the notion of Kernel Methods needs to be introduced. Kernel Methods are the main component of SVMs, and are the reason why SVMs became so popularized with respect to solving many different types of classification problems. Kernel Methods allow the data set to be projected into higher dimensions without losing information. This is also referred to as the Kernel Trick.

To explain the kernel mapping: Let  $\mathcal{X}$  be a data space, and  $x \in \mathcal{X}$ .  $\exists$  a feature map  $\phi$  which maps the elements  $x$  into a higher dimension called a Hilbert Space denoted as  $\mathcal{H}$ . The kernel of interest, denoted  $K(x, w)$  is computed as the inner product  $\langle x, w \rangle_K = \langle \phi(x), \phi(w) \rangle_{\mathcal{H}} = K(x, w)$ . This kernel replaces the inner product between the data vectors. For the mapping to work,  $K(x, w)$  must satisfy three conditions:

1.  $K$  is continuous
2.  $K$  is symmetric
3.  $K$  is positive definite

If  $K(x, w)$  satisfy these conditions, then  $K(x, w)$  is said to be a Mercer Kernel. This Mercer Kernel defines a Hilbert Space  $\mathcal{H}$ , and a feature map  $\phi : \mathcal{X} \rightarrow \mathcal{H}$  s.t.  $\langle \phi(x), \phi(w) \rangle_K = K(x, w)$ .

For the Kernel Method to be useful there must be a mapping for the data set specified such that the classification can be applied to the projected data with more ease than without using Kernel Methods. To be able to implement Kernel Methods it is required to replace the inner product between the data vectors with a specified

Kernel Function. This will allow for projection of the data vectors into higher dimensions. Furthermore, a linear classifier could be applied onto the projected data to discriminate between the different labels.

Data that is linearly separable can be classified very quickly. Unfortunately, the vast majority of data encountered in the world not linearly separable. By using the Kernel Method, the data is transformed into a higher dimensional space where it becomes linearly separable. This allows the linear classifiers to solve far more problems than if it were in a lower dimension. An example of this can be seen in **[Figure 6]** taken from **[3]**. In this figure, the black and white dots originally cannot be linearly classified, but when using Kernel Methods and projecting data into a higher dimension, the data is indeed able to be linearly classified. It should be noted that although the decision is linear in the high-dimensional feature space, it need not be linear in the original space the data occupied.

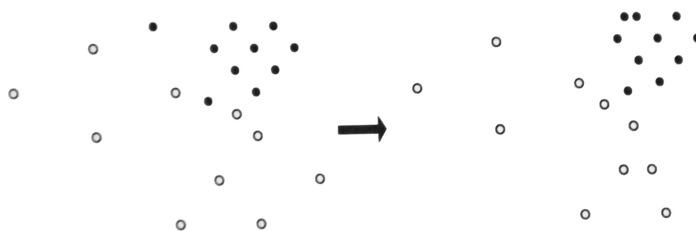


Figure 6: SVM Example

Another example of the Kernel Method can be seen in **[Figure 7]**. With respect to this figure, in  $n$ -dimensional space, a hyper plane is a subset of dimensionality  $n - 1$ . Recall, kernels project the data into a high (possibly infinite) dimensional feature space. Working with infinite dimensional hyper-planes is not necessarily difficult, but it is difficult with respect to classification. See **[3]** for more details concerning this matter.

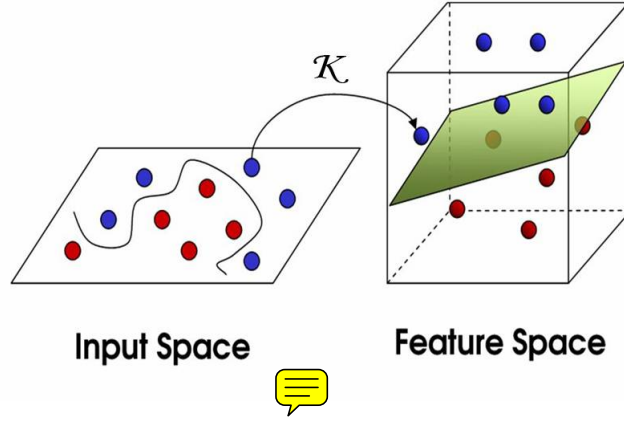


Figure 7: Kernel Method

For the two and four key classification problems the introduction of the terms: supervised learning, training data and test data are necessary. SVMs are a form of supervised learning. They use labeled training data to learn the classification rule for the separating hyper plane. This classification rule is then evaluated against test data. If  $X$  is the Training Data, and  $Y$  is the labeling system, the SVM will then read in the paired values of  $(X_i, Y_i)$  and store them into memory, where  $i$  is from 1 to the number of key hits. The labeling of training data will entail attaching the name of each key to its vector of characteristic features, in other words if the isolated signal comes from the A-key, the label “A” will be placed in the vector  $Y$  correspondingly. Some of these data points will serve as the support vectors which will define the separating hyper plane. This classification rule will then be fed test data to assess its’ accuracy. Essentially, it is given new  $X_i$  and attempts to classify them as the appropriate  $Y_i$  based on the decision rule, which was learned in the previous step. Comparing the predicted labels with the true labels provides a measure the accuracy of our SVM.

For an example, consider the one-class problem when applied to the SVM. The training data could be considered in this case: 10 key strokes of the A-key. The accuracy of the SVM will be measured with the predictability of the SVM labeling the

test data with the true labels  $Y$  that were fed into the SVM. If running the SVM with one class, this is denoted as Novelty Detection. The results will show any outliers from the ordinary data set being classified. For example if the test data set had 9 key strokes of key-A and one key stroke of key-L, the SVM will not know how to handle the new L data. The L key will be detected as an outlier within the summary of statistics in the SVM model.

Now the data is ready, and there is enough motivation and background information to move into the problem that was posed in this thesis; The Two Key Classification Problem.

## 5 The Two Key Classification Problem

For the two key classification problem, this thesis shows the discrimination between the A and L keys, by using labels and the sound generated from the keystroke. For a typical binary classification problem there is a data set  $X$ , and a labeling system  $Y$ . The goal is to find a prediction for  $Y$  that will be able to match the true labeling of  $Y$ . A great advantage of methods like SVMs is their speed. The downside of SVMs is that it is difficult to solve multi-classification problems, in which case the SVMs will use the one-versus-all method of binary classification to come to a conclusion. This will be shown in the four key discrimination problem. SVMs have been implemented directly to this type of problem where the data set  $X$  is large and there are two or more classes.

The data set in this two key scenario is the feature matrix extracted from the isolated signals of the A and L key. For this problem the A and L key are both hit 50 times. These raw signals are very large, and thus needed to be reduced using the technique of extracting the most prominent features. After obtaining the raw signals of the A and L key sequence, they are sent through the GH function. The GH function isolates the signals in preparation for extracting the most important features using MFCCs and autocorrelation coefficients. The raw signals of the A and L key are combined into a 100x600 matrix by concatenation. This will be called the matrix of naked signals or  $X$ . The feature matrix is prepared by first taking the autocorrelation coefficients of the naked signals matrix  $X$ . Recall that 65 autocorrelation coefficients will be extracted from each individual isolated signal and placed into a new matrix called FeaturesX. FeaturesX has a dimension of 100x65 and has reduced the data space drastically.

The next step is to obtain the MFCCs of the naked signal matrix. The MFCC

function can only be used on data of the class wave, thus the matrix of naked signals is transformed into a wave. After transformation into a wave, the MFCC function applied to the naked signal will result in a number of coefficients dependent on the 7 parameters of the function. The number of coefficients is approximately less than 50, but will vary with each iteration later when trying to optimize the accuracy of the SVM. For the initial parameters selected there are 21 MFCCs produced. When combined with the autocorrelation coefficients a 100x86 new features matrix is produced and ready to be read into the SVM for analysis. This new features matrix filled with the altered A and L data will be the training data. The labeling system is created by imposing the 50 labels of “A” to each respective isolated A signal, and 50 labels of “L” to each respective isolated L signal. Now that there is a labeling system and training data, the SVM can be implemented to be trained.

The SVM has no problem discriminating between the A and L key, and the image of the SVM model can be seen in [Figure 8]. The R code associated to the results can be seen below:



```
> # Assign the labels/Set dimensions for Y (1x100)
> # rep(want to replicate,number of times to replicate)
> Y = as.factor(c(rep("A",50),rep("L",50)))
> dim(y) = c(100,1)
> # Classify with SVM
> model = svm(TrainingData,Y,scale=SCALE)
```

Call:

```
svm.default(x = TrainingData, y = Y, scale = SCALE)
```

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

```

Number of Support Vectors:  43

> #Predict and Check

> pred0 = predict(model,TrainingData, decision.values=TRUE)

> #Results of classification of A against L

    #Classified 50 A's as A, and 50 L's as L

> table(pred0,Y)

      y
pred0 A  L
     A 50  0
     L  0 50

```

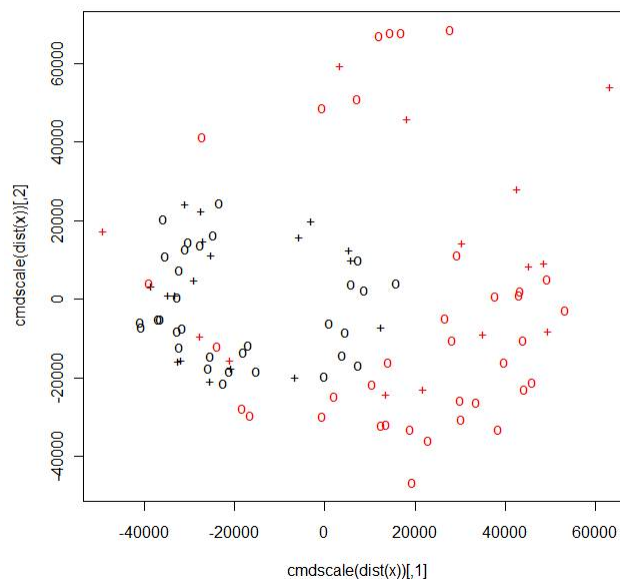


Figure 8: SVM Binary Classification A-Key L-Key

This figure shows that the SVM uses 43 support vectors, marked as x's. The circles represent the two keys being classified. These support vectors are the points which sit on the margin of the maximum-margin hyper-plane. The maximum-margin hyper-plane is the optimized hyper-plane which is projected into a higher dimension

to linearly classify the A-key and L-Key. This can be seen in [Figure 9]. The maximum-margin hyper-plane is of the class:

$$\text{sgn}(w \cdot x + b) \quad w \in \mathbb{R}^d, b \in \mathbb{R} \quad (1)$$

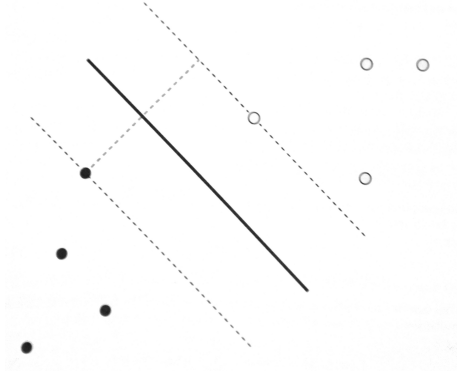


Figure 9: SVM Binary Classification A-Key L-Key

In order to optimize this maximum margin, (1) needs to be optimized. To optimize this equation it is necessary to use a LaGrange function and LaGrange multipliers [3], which will not be covered in this thesis. The lower amount of support vectors that are needed to classify the data with respect to the total amount of support vectors, the more predictive power the SVM model will have on future testing data. Therefore, 32 support vectors out of a possible 100 will be shown to have good predictive power on new test data. If all 100 support vectors were used, then the SVM would be nothing more than a memory machine and would not have good predictive power on new test data. Now that there is a trained SVM model, the next step is to test new data with the SVM. The new data tested was a 45 character random sequence of A's and L's.

The same procedure of obtaining the features matrix of this test data is used. This signal needs to be isolated into the hits using GH. Application of the autocor-

relation function gives a 45x65 matrix, which extracts 65 autocorrelation coefficients for each isolated signal. Transformation of the naked signal matrix into a wave allows for the MFCCs to be extracted. With the initial parameters there are 21 MFCCs associated to each of the individual isolated signals. After combining the autocorrelation coefficients and the MFCCs, the test data matrix is now complete. The last thing needed is to create the true labels for our test data matrix. This is a matrix of the string of characters below denoted ytrue. The goal for the SVM is to find a prediction for the labeled training data that will be able to match the true labeling of the test data. The R code for the results can be seen below:

```
> #Predict and Check
> pred = predict(model,TestData, decision.values = TRUE)
> # True labels
> ytrue = c("L", "A", "A", "L", "A", "L", "A", "A", "A", "L",
+ "L", "L", "L", "L", "L", "A", "A", "A", "A",
+ "A", "L", "A", "A", "L", "A", "A", "A", "L",
+ "A", "L", "L", "A", "A", "A", "A", "L", "A",
+ "L", "L", "A", "L", "A", "L", "A", "L")
> table(pred,ytrue)

      ytrue
pred  A  L
  A 25  0
  L  0 20
[1] "45 out of 45. i.e. 100 % correct"
```

The end result of this problem was that the A-key and L-key were able to be perfectly discriminated against within 20 iterations of the procedure listed above. To get 100% accuracy in prediction of the test data it was necessary to optimize the parameters for the MFCC function. The MFCC function has 7 parameters that can

be set. Therefore, a loop which sampled uniformly around each parameter with a  $\pm dt$  specific to each parameter was implemented. This iterative procedure evaluated each new set of parameters with the MFCCs and produced the accuracy of the modeled SVM until it reached 100%. This two key classification problem is nothing more than a binary classification problem. This iterative procedure produced fast results, and was able to be generalized to incorporate more keys to be classified. This will be explained in the Four Key Classification Problem.

## 6 The Four Key Classification Problem

The Four Key Classification Problem uses the same techniques as the Two Key Classification Problem. The introduction of the Q and P key was made into the analysis along with the A and L key, with 50 key hits recorded respectively for each key. The main goal is to classify four keys using MFCCs and autocorrelation coefficients, and obtain at least an 80% accuracy rate for the SVM. The SVM is not able to properly classify all four keys at once. Therefore, the method that will be used to get results for this problem is multiple binary classifications of the four keys. The multiple binary classifications will go as follows:

1. Discriminate A-key from all other keys  $\rightarrow$  A against not A
2. Discriminate L-key from all other keys  $\rightarrow$  L against (not L  $\cap$  not A)
3. Discriminate Q-key from all other keys  $\rightarrow$  Q against (not Q  $\cap$  not L  $\cap$  not A)
4. Three binary decisions are all that will be needed in this problem, because the last key, P, will be classified as whatever is left over from the first three steps.

$$A + L + Q + P = \text{Total Number of Keys}$$

$$P = \text{Total Number Of Keys} - A - L - Q$$

The following analysis is a single example, with one specific set of parameters. To obtain the training data set  $X$ , 200 isolated signals are extracted from the four sets of 50 key hits. These isolated signals are concatenated into one matrix. The labeling of this data will be exactly the same as the procedure in the two key problem, where each key will have a label referencing the key being hit. This label matrix will be denoted as  $Y$ . The autocorrelation function is used with the same lag as the two key problem producing 65 coefficients for each isolated signal. The dimension of the autocorrelation matrix for the training data is then 200x65. The MFCCs extracted

from each isolated signal is increased from the two key classification. On average approximately 250 more MFCCs are used in this scenario. The properties, and number of MFCCs extracted is dependent on the 7 parameters of the MFCC function. They are changed with each iteration of the classification procedure to search for optimal combinations of parameters which allow the MFCC's to contain more information about the waves. The MFCCs and the autocorrelation coefficients are concatenated into a single matrix, which are used as the features of the isolated signals in the following analysis.

The analysis which follows is different from the two key problem's. Starting with the A-key and using the label matrix  $Y$ , the first binary classification will be testing the A key against not the A-key. In other words a new label matrix denoted  $Y_A$ , contains the values "A" if the key is an A or 0 otherwise. The first binary classification uses the features matrix and the labeling  $Y_A$ . The SVM is now trained with the paired values of  $(X_i, Y_{Ai})$  into its' memory. The index  $i$  ranges from 1 to 200. Following this step the test data set needs to be prepared. A random sequence of 45 Q and P key hits are recorded, and are then concatenated with the recorded random A and L keys to make the raw test data matrix. This raw test data matrix has a dimension of 90x600. The features are extracted from this matrix using the same techniques, and produce the training data matrix consisting of the MFCCs and the autocorrelation coefficients. The labeled training data is as follows:

```
> ytrue =c("L", "A", "A", "L", "A", "L", "A", "A", "A", "L",
+          "L", "L", "L", "L", "L", "A", "A", "A", "A",
+          "A", "L", "A", "A", "L", "A", "A", "A", "L",
+          "A", "L", "L", "A", "A", "A", "A", "L", "A",
+          "L", "L", "A", "L", "A", "L", "A", "L", "P", "Q", "Q", "Q",
+          "Q", "Q", "Q", "Q", "P", "P", "Q", "Q", "Q",
```

```

+      "Q", "Q", "Q", "P", "P", "P", "P", "P", "P", "Q", "Q", "Q",
+      "Q", "Q", "Q", "P", "Q", "Q", "Q", "Q", "P", "Q", "P", "Q",
+      "P", "Q", "P", "P", "P", "P", "P", "Q")

```

The same type of labeling is used for the random sequence of characters where the character is labeled as an “A” if it is truly an A and 0 otherwise. This sequence is denoted as  $y_{trueA}$ , and will be used in evaluation of the accuracy with the labels of the training data  $Y_A$ . The SVM model is now checked with the test data. Based on the results, the SVM was able to correctly classify 25 A keys while incorrectly classifying 14 A keys. Therefore, after the first binary classification there are 51 keys left able to try and optimize the accuracy of the SVM. Thus the test data set is now reduced to 51 or  $90 - 25 - 14 = 51$ . The output from R can be seen below:

```

> # Classify A against not A with SVM
> model = svm(ACFMFCC2x,yA,scale=TRUE,type = "C-classification")
Number of Support Vectors: 134
> #Predict and Check
> pred0 = predict(model,TrainingData, decision.values=TRUE)
> table(pred0,yA)

```

	yA	
pred0	0	A
0	150	0
A	0	50

To continue analysis the next binary classification will be assessing the L key against all other keys. This classification does not include any keys which were classified as A in the previous step regardless of whether the svm correctly labeled the test data or not for these keys. They are removed accordingly. This new subset consisting of 51 isolated signals must have their features extracted to create the new test data.

Now the same procedure is used as the A key to train the next SVM with L against all keys that are not L. This includes the full training data and the labeling produces a matrix denoted  $Y_L$  with “L” if the true value is an L and 0 otherwise. To clarify, the dimension of  $Y_L$  is  $200 \times 1$ . Now the SVM is trained with the original feature matrix and the label  $Y_L$ . The SVM model is used with the new test data of the 51 isolated signals to assess the accuracy in classifying the L’s. 4 L’s are classified correctly, while 23 L’s are classified incorrectly. This leaves a total of 24 keys left able to try and optimize the accuracy of the SVM. Thus the test data set is now reduced to 24 or  $51 - 23 - 4 = 24$ .

One more binary classification is needed to assess the accuracy of the overall classification for the four key problem. This last binary classification tests all keys that are left against the Q key. The same procedure is used with the test data set for classification; which is now reduced to 24 isolated signals. The result of this binary classification predicts 0 Q keys correctly and 1 Q key incorrectly. There are 23 keys left to try and optimize the accuracy of the SVM. These 23 keys are exactly the number of P keys that were predicted correctly. Therefore, final assessment can be calculated as follows: The performance of the overall classification is the summation of correctly predicted keys within each SVM model divided by 90, and lastly multiplied by 100 to get an initial percentage. This is  $(25 + 4 + 1 + 23)/90 \cdot 100 = 58.89\%$  to two decimal places. The final SVM plot can be seen in **[Figure 10]**. In other words 52 out of 90 keys were classified correctly.

After over 100 iterations of this procedure the model was not able to optimize the classification of these four keys past 65.55%. In other words 59 keys out of 90 were able to be classified with multiple iterations of the four key binary classification procedure. In general the SVM should normally attain at least 80% accuracy. The

SVM was unable to reach this number and therefore it is natural to assume that there is room for improvement, and it is not being fed the best features. As more keys were added to the model, there was a decrease in the number of keys left to classify. This means that the accuracy of the model will drastically decrease as the number of keys increases. With adding two keys to this model there was a decrease of 34.45% in accuracy.

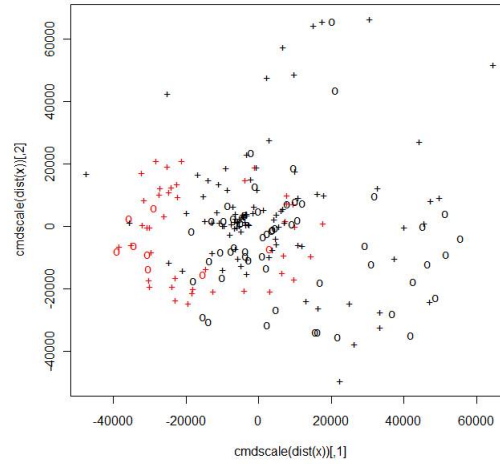


Figure 10: SVM Binary Classification A-Key L-Key Q-Key P-Key



## 7 Conclusion

When going from the Two Key Classification Problem to the Four Key Classification Problem, the results are drastically different. This is suggestive that the features that are being used to model the isolated signals could be improved. [6], discusses how to improve the way that features are being extracted from the isolated signals. While MFCCs along with autocorrelation coefficients generate advantageous results with respect to random chance, the MFCCs are especially used for speech recognition. From [6], it is shown that the intensity function of the MFCC is not being used to it's fullest extent. It derives an intensity function which can more accurately model the signals produced from a keyboard.



## References

- [1] Agrawal, Rakesh, and Asonov, Dmitri, “Keyboard Acoustic Emanations,” *IBM Almaden Research Center*, pp. 1-9, 2004.
- [2] Meadows, Catherine, and Paul Syverson, “Keyboard Acoustic Emanations Revisited,” *Proceedings of the 12th ACM Conference on Computer and Communications Security* pp.373-382, New York: ACM, Nov. 2005.
- [3] Camastra, Francesco, and Alessandro, Vinciarelli, “Machine Learning for Audio, Image and Video Analysis: Theory and Applications,” London: Springer, 2008.
- [4] “Fourier Transform,” *Wikipedia, the Free Encyclopedia*, “[http://en.wikipedia.org/wiki/Fourier\\_transform](http://en.wikipedia.org/wiki/Fourier_transform)”.
- [5] “Discrete Cosine Transform,” *Wikipedia, the Free Encyclopedia*, “[http://en.wikipedia.org/wiki/Discrete\\_cosine\\_transform](http://en.wikipedia.org/wiki/Discrete_cosine_transform)”.
- [6] Jarrett, Nicholas W. “Not MFCCs?!” Thesis. University At Albany, State University of New York, 2010. Print.
- [7] <http://accn.net/pmwiki/pmwiki.php/Keys/R>. Carlos Rodriguez. Web.