6.867 Machine Learning Final Project
Classification of 2D gestures using Hidden Markov Models and Support Vector Machines

Enrique Fernandez
[redacted]@mit.edu
Massachusetts Institute of Technology
December 9, 2012

Abstract

In my final project for 6.867 Machine Learning I present a method for classifying two dimensional gestures using two different methods (HMMs and SVMs). I describe the type of gestures under consideration and detail the experimental setup that I used to capture the gestures. Then I provide a method to extract features from the recorded two dimensional gestures that is suitable for classification tasks. Next I describe a method for gesture classification that achieves accuracy rates of **96.81%** (for test examples of trained gestures) and **96.2%** (for test examples of trained gestures, non-trained gestures and random sequences). Similarly, I provide a classification method using SVMs that achieves rates of **94.2%** and **93.8%** under the same conditions. Finally, I conclude summarizing my findings and comparing both methods for this task.

1 Introduction

In a world in which smart phones and tablets with touchscreens are present everywhere, the ability of detecting gestures done with our fingers in screens has become ubiquitous and doesn't even surprise us anymore. For my Fall 2012 Machine Learning Final Project, I intend to understand how the machine learning techniques that we've learned throughout the semester can be applied to recognize two dimensional gestures and what kind of techniques are better suited for this purpose.

To fulfill my objective, I focus on the task of recognizing gestures done with either a mouse or a touchpad on an interface window that I developed in MATLAB. The captured gestures are then processed to extract the relevant features and then Hidden Markov Models and Support Vector Machines are used to classify the gestures.

My paper starts with a description of the types of gestures studied in this project and follows with my approach to extract relevant features from those. Then the next section focuses on the use of Hidden Markov Models for gesture classification. I describe how to approach this problem using HMMs, how to train the models and the results obtained. I also comment the problems that arise from using HMMs for this type of task.

After discussing the HMMs results, I move on to discuss the use of Support Vector Machines for gesture classification. In this section I revisit the gesture features discussed previously given the limitation of needing to have a fixed number of features and propose a method for assembling a fixed length feature vector that is suitable for SVM classification. After that, I discuss how to apply SVMs to this problem, the results obtained and the limitations faced.

Finally, I compare both methods for classifying gestures and I end my paper with a summary of the lessons learned and concepts used while working on this project.

2 Two dimensional gestures

As mentioned before, the focus of this project is to classify gestures that the user inputs using the mouse or a trackpad. In this section I intend to describe how those gestures are defined and processed.

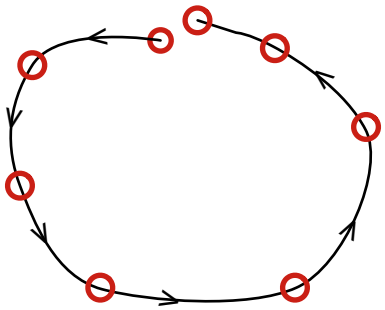

Figure 1: This diagram represents gesture 'O'. The black line indicates the path followed by the user whereas the red circles indicate the sampled points stored. The arrows indicate the direction in which the path was followed.

Gestures are defined by the path followed by the user

with the mouse. While the user executes a gesture, the gesture recording interface that I wrote records the cursor positions and times periodically as three dimensional vectors in the form (x_i, y_i, t_i). I decided to sample the points at *30 Hz* after recording several gestures and determined that this number constituted a good compromise between the amount of information recorded and complexity. Figure 1 shows an example of the circle gesture (represented as "O"). Note that not only the path followed is important, but also the direction in which it was followed.

2.1 Extracting the features

A two dimensional gesture described by the sequence of three dimensional vectors in the form (x_i, y_i, t_i) as in section 2 is completely defined, in the sense that it contains all the information required to reconstruct the gesture. However, this encoding of gestures may not be the best one for classification purposes. For our application, we want to define the best set of features that can be used to differentiate between gestures. For that purpose, it is important that different examples of the same gesture have associated similar features. It can easily be seen that the original encoding of three dimensional vectors is not a good choice because the vectors highly depend on the position where the gesture was executed and on the scale of the gesture.

There are many possible choices for two dimensional gesture features that we could try such as velocity between sampled points, relative positions from the center of the gesture, etc. In this project, I have chosen as features the sequence of angles between sampled points. The angle between two consecutive sample points can be calculated using the following formula:

$$\alpha_i = arctan \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad (1)$$

The angles calculated using (1) are then discretized in 16 *codewords* that correspond to bins of 22.5°. The gesture encoding is finally a sequence of discretized angles represented by one of the 16 codewords. Note that the length of the sequence is not fixed and depends on the duration of each gesture example. Figure 2 represents how this discretization is done.

There are several reasons why I have chosen this particular encoding. In first place, the gestures represented in this way don't depend on the absolute positions anymore. Moreover, this approach has been used successfully in academia. (Dadgostar et al. 2006), (Lee, Kim, and on 1999), (Elmezain et al. 2008) and (Liu, Lovell, and Kootsookos 2004) among others report positive results when using this encoding.

Note that under this encoding, the problem of classifying gestures is reduced to the problem of comparing signals of different lengths. Figure 3 shows the discretized sequences of four examples of the circle gesture. We can see how, although all sequences have different lengths, they seem quite similar.

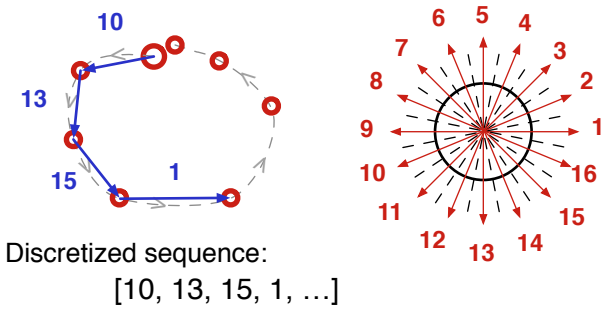

Figure 2: The angles between the sampled points are calculated and then discretized in 16 categories (or codewords) as indicated in the diagram above. An example of how a circle gesture is converted to a sequence of discretized angles is shown too.

2.2 Studied gestures

In this project I defined 11 gestures. These gestures can be found in table 1. The table shows the representation of each gesture as well as its numerical representation and short name. These gestures are referenced using this number or short name throughout this paper.

3 Experimental setup and data collection

All the data used in this project was captured specifically for this purpose and no preexisting databases were used. In order to capture the gestures, I developed a MATLAB interface that recorded sequences of three-dimensional vectors (x_i, y_i, t_i) at a sample rate of *30 Hz* (Figure 4). This interface allowed the users to record gestures by clicking with the left mouse button and dragging until the button is released.

Using this system, two subjects recorded 60 repetitions of each of the 11 gestures, resulting in a total of 120 examples per gesture and 1320 examples of all gestures. These examples were then converted into the discretized angle sequences as described in section 2.1.

The data was then divided into training and testing data sets (50%). It is important to mention that this division was done randomly in order to attenuate the differences between the two subjects and inside gesture series recorded by each user (a user could start executing a gesture in some way in the beginning of the gesture series and ending in a different way). After this step the data is finally ready to be used to train the models.

4 Hidden Markov Models

Hidden Markov Models are a special kind of Bayes Networks that consist of states and observations. The state at a given time $i + 1$ only depends on the state at the

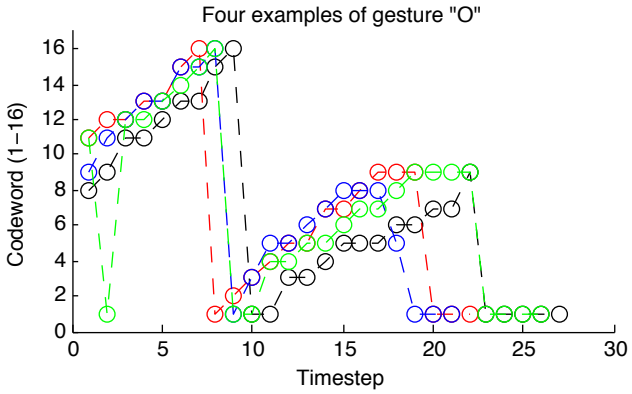

Figure 3: This figure shows the discretized sequences of four examples of the circle gesture. The x axis represents the time steps at which the gesture was sampled and the y axis represents the discretized angle codewords. Note that examples have different lengths.

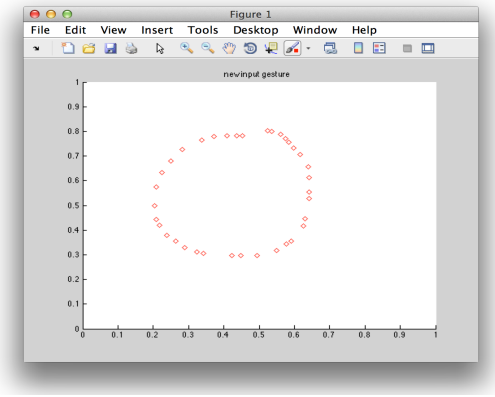

Figure 4: MATLAB interface for capturing gestures.

previous time i (Markov process). However, in Hidden Markov Models (HMMs) the states are not observable and only the outputs of these states can be observed. The outputs of a state at time i (observation) only depend on that state. A HMM of N states and M observations is fully defined by the parameters:

$$\lambda = (\pi, A, B) \tag{2}$$

where π is a $N \times 1$ vector of initial state probabilities, A is the $N \times N$ transition probabilities matrix and B is the $N \times M$ emission probabilities matrix.

HMMs are very often used in applications in which the data is sequential or depends in previous data. HMMs have been broadly applied in speech recognition and in biology applications such as DNA sequencing.

Given that the two dimensional gestures that we are studying in this paper are sequential in nature and that they can be understood as series of states that need to be followed to complete a gesture, it seems very reasonable to try to model them as Hidden Markov Models.

#	Name	Symbol
1	L	
2	O	
3	V	
4	Z	
5	M	
6	G	
7	T	
8	And	
9	GT	
10	W	
11	C	

Table 1: Table of gestures.

In fact, researchers have successfully used HMMs in the past to classify gestures. One of the most famous applications of HMMs in this area was the automatic recognition of the American Sign Language by extracting the gestures from videos (Starner and Pentland 1995). Others have also used HMMs to classify gestures from videos (Lee, Kim, and on 1999), (Elmezain et al. 2008), (Liu, Lovell, and Kootsookos 2004).

4.1 Gesture classification using HMMs

Framing the two dimensional gestures as defined in section 2 as HMMs is pretty straightforward. We can understand the discretized angle sequence (defined in section 2.1) as a sequence of observations arising from an HMM model for which the corresponding state sequence is not known. This is represented in Figure 5.

Now, using the training data for each of the 11 gestures, we can train an HMM model for each using the Baum-Welch algorithm as will be discussed later. Then,

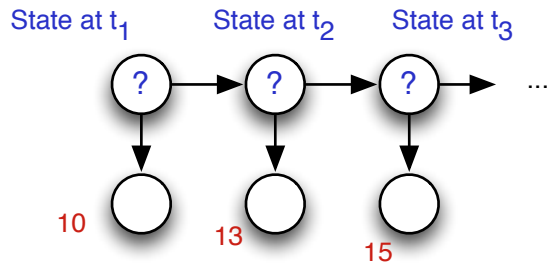

Figure 5: Representation of gestures using HMMs.

in order to classify an unseen gesture, we can compute the log likelihood of the discretized sequence of angles for each of the HMM models. We finally select the model that gives the highest likelihood for that sequence.

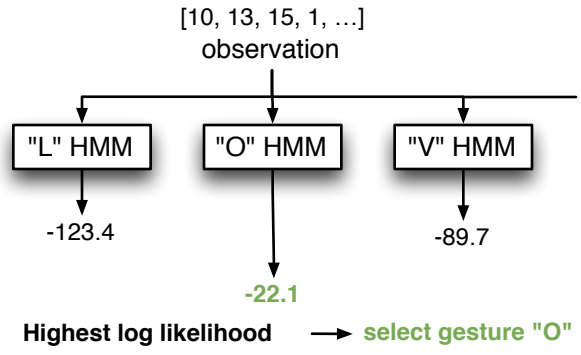

Figure 6: The model selected is the one that achieves the highest log likelihood.

4.2 Selecting the number of states

Given the training data consisting of a set of observations (sequences of discretized angles) for each gesture, the Baum-Welch algorithm can provide the parameters of the HMM model $\lambda = (\pi, A, B)$ provided that we specify the number of states of the HMM. It is therefore clear that selecting the number of states is an important aspect of learning HMM models and that we need a way of choosing it for each of the 11 gestures we intend to train. Apart from that, we can also choose the type of transition matrix that we want to use in our model. A HMM can be *ergodic*, if any state can be reached from any other state in a finite number of steps, *Left-Right (LR)* if transitions are only allowed between a state and the following ones (the transition matrix is 0 below the diagonal) or *Left-Right Banded (LRB)* if the only transitions allowed are staying in the same state or transitioning to the immediate following one.

In order to select the number of states for each gesture and the type of HMM model (*ergodic*, *LR* or *LRB*), I used 5-fold cross-validation. I specified model candidates with number of states ranging from 4 to 12 and different transition matrix types. Those candidates were then ranked according to the average likelihood per gesture that they achieved in the testing k-fold. Table 2 shows an example of using cross-validation on the training data for gesture "L". In this example, the best model for "L" is the *LRB* with 7 states. Note however that the average likelihood doesn't vary too much as we change the number of states.

Cross-validation for gesture "L"

n_{states}	LRB	Ergodic
4	-9.0596	-9.6064
5	-8.5025	-9.6763
6	-8.7514	-9.4284
7	**-8.4314**	-9.2835
8	-8.7470	-9.2723
9	-8.7752	-9.2878
10	-8.6707	-9.2192
11	-9.0334	-9.3470
12	-8.5364	-9.0578

Table 2: This table shows the results of 5-fold cross-validation on the training data of gesture "L". Note that in this case the best model is the *LRB* with 7 states.

The same procedure was applied to all the other gestures, but is not shown here for brevity reasons. Regarding the type of transition matrix, in my tests *Left-Right Banded (LRB)* models performed slightly better than *Left-Right (LR)* models and significantly better than *ergodic* models consistently. This result is not surprising given that we represent the gestures as successions of states and that once we reach a state we shouldn't move back.

Also, (Liu, Lovell, and Kootsookos 2004) conducted a study to assess the performance of HMM models for gesture classification depending on the type of transition matrix. They conclude that *LRB* models perform in general better than *ergodic* and are preferred for this type of applications. Due to this and to my preliminary results, I chose to use *LRB* HMM models to classify the two dimensional gestures.

Table 3 shows the final number of states used for each gesture HMM. Note that because very often the differences in average likelihood achieved by different number of states was very low, the final chosen number of states is not always the one that achieved the best score, but the one that represents a good compromise between a good score and complexity.

Number of states per gesture HMM

#	Gesture	n_{states}	threshold
1	L	7	-30
2	O	8	-45
3	V	8	-25
4	Z	9	-30
5	M	9	-40
6	G	8	-35
7	T	5	-25
8	And	7	-40
9	GT	7	-35
10	W	10	-45
11	C	7	-32

Table 3: This table shows the number of states chosen for each HMM (LRB) resulting from 5-fold cross-validation. The threshold column for each gesture indicates the minimum likelihood that a new unseen sequence needs to exceed in order to be eligible to that gesture.

4.3 Implementation

In order to successfully use Hidden Markov Models to classify two dimensional gestures there are two main things that we need to be able to do:

1. Calculate the likelihood of an observation sequence given the HMM parameters. $P_i(O|\lambda_i)$
2. Calculate the maximum likelihood estimates for the HMM parameters given a set of observations (training data)

The first problem can be solved efficiently using the forward algorithm (equivalent to the message passing algorithm that we've studied in class). For the second problem, there's no known way of achieving the maximum likelihood estimates, but it is possible to get to local maximums using an iterative algorithm that depends on the initial parameters. The most popular algorithm for this is the *Baum-Welch* algorithm, which is a particular case of the Generalized Expectation Maximization (GEM) algorithm.

In order to focus on the more interesting areas of this project and given the scope and time frame, I chose to use an existing HMM library that provided me with methods to solve these two problems. The library I used is called the *Hidden Markov Model Toolbox for MATLAB* and is authored by Kevin Murphy. This library can be found at http://www.cs.ubc.ca/ murphyk/Software/HMM/hmm.html.

To train the HMM models, I initialized the HMM parameters with the following constraints:

1. The initial state is always the first one ($\pi_1 = 1$ and $\pi_k = 0$)
2. The transition matrix A is an stochastic matrix initialized at random with the constraint that all the elements except for $a_{i,i}$ and $a_{i,i+1}$ are zero (we are using LRB HMM models).
3. B is an stochastic matrix initialized at random.

Once these parameters are initialized, the HMM Toolbox for MATLAB can estimate local maximum likelihood parameters using the Baum-Welch algorithm, although they depend on the random initial conditions.

4.4 Classification results

Using the training data collected as described in section 3 (60 training examples for each of the 11 gestures) and following the procedure described in the previous sections, the 11 HMM models were trained. These models were then tested using the test data (a different set consisting of 60 examples per each of the 11 gestures). The results are presented in Table 4.

HMM Classification results

#	Gesture	Tests	Correct	Errors	%
1	L	60	60	0	100
2	O	60	59	1	98.33
3	V	60	50	10	83.33
4	Z	60	59	1	98.33
5	M	60	59	1	98.33
6	G	60	60	0	100
7	T	60	60	0	100
8	And	60	57	3	95
9	GT	60	57	3	95
10	W	60	60	0	100
11	C	60	58	2	96.67
	Total	**660**	**639**	**21**	**96.81**

Table 4: This table presents the results of testing the trained HMMs for each gesture on the testing data.

As it can be seen in the table, the results are in general very good and HMMs achieve a global classification rate of 96.81% (639/660). However, it is important to note that one of the gestures yields a much worse performance than the others. Gesture "V" is correctly classified only 83% of the times, more than 10 points worse than the other gestures. Careful analysis of the data indicates that the problem with gesture "V" is that it is being classified as "W" sometimes. This happens because "V" can be understood as doing half the "W" gesture. Because we are not using fixed sequences, the "W" assigns a high likelihood to the "V" gesture because it is "going in the right path".

4.5 Problems encountered

Although we've seen that HMMs provide very good results for classification of two dimensional gestures, there are some limitations in the procedure described so far that needs to be highlighted.

0 likelihood
Given that the gesture examples are converted to discretized sequences of codewords, it sometimes happens that there is an example of a gesture that includes a

codeword that had never been seen before in the training examples. When this happens, the example gets assigned a log likelihood of $-\infty$. This can happen when a gesture is a little more deformed than the others and one of the angles of the sequence falls into a bin in which no other point had fallen before. Although it is logical that this happens, this problem can be hard to deal with in certain cases, such as when we do cross-validation. My approach in this project has been to eliminate the problematic example during cross-validation training because this situation rarely happens and the example will be considered anyway during final training after the number of states is decided.

Untrained gestures and random sequences
The results shown in Table 4 are very good, but it is important to realize that they only apply to the case in which the input unseen gesture is necessarily one among the trained ones. Therefore, in order to classify the new gesture we only need to select the HMM model that yields the highest log likelihood as explained in section 4.1. It is interesting to consider how to extend the system proposed in this paper to handle unseen gestures and random sequences in a way that they would be classified as *unrecognized* as opposed to one of the trained gestures.

In order to do that, I propose to use *thresholds*. Now, in order to classify a new observation as a trained gesture it is no longer sufficient that the observation achieves the highest likelihood for that model, but it also needs to be bigger than a certain threshold that is different for each HMM model. If an observation doesn't achieve any likelihood bigger than its corresponding threshold, then the observation is classified as an *unrecognized gesture*.

In order to select the thresholds for the HMM models, I used cross-validation again. I chose the thresholds as lower bounds of the minimum average likelihoods achieved during cross-validation. Note that these thresholds do not need to be very precise as its mission is to reject candidates that achieve some likelihood in the model but lower than what a real gesture of that model would. The thresholds that I chose for the HMM models are shown in Table 3.

Table 5 shows the results of using thresholds for gesture classification with HMMs. In this case, I only trained the first 9 gestures. Then, the test set was assembled with 60 examples from the testing set of each of the 9 gestures plus 60 randomly generated sequences and 60 examples of gestures "W" and "C", that were not trained. The examples for the random sequences, and gestures "W" and "C" are considered correctly classified if they are assigned the class *unrecognized gesture*.

As we can see in the table, using thresholds works reasonably well and allows us to differentiate between known and unknown gestures. However, it is expected that the threshold system will start to perform worse as the number of gestures increases and more alike gestures are defined.

HMM Classification results

#	Gesture	Tests	Correct	Errors	%
	Trained				
1	L	60	60	0	100
2	O	60	59	1	98.33
3	V	60	56	4	93.33
4	Z	60	56	4	96.33
5	M	60	55	5	91.67
6	G	60	59	1	98.33
7	T	60	60	0	100
8	And	60	52	8	86.67
9	GT	60	57	3	95
	Random				
	Random seq	60	60	0	100
	Untrained				
10	W	60	60	0	100
11	C	60	59	1	98.33
	Total	**720**	**693**	**27**	**96.2**

Table 5: Testing results of HMM models using thresholds. Gestures "W" and "C" were not trained and classification is considered correct if the corresponding examples were assigned the unrecognized class. The same applies to the random data.

5 Support Vector Machines

The purpose of this section is to assess the performance of Support Vector Machines in the task of classifying two dimensional gestures. As we've discussed in the previous section, Hidden Markov Models are a good fit for this task as they consider the evolution of the states of the gesture and take into account time sequences. Now in this section I describe how to use SVMs for the same purpose.

5.1 Revisiting the features

We discussed in section 2.1 that a good way to encode the recorded gestures to use with HMMs was to compute the sequence of discretized angles. As discussed back then, these generated sequences had varying lengths depending on how long it took to execute the gesture. That was not a problem then as given an HMM, it is always possible to compute the likelihood of a sequence of observations of any arbitrary length.

Unfortunately, the same doesn't apply with SVMs. In order to use SVMs we need to define a fixed set of features that needs to be the same for all training and testing examples. When working on the problem of deciding how to adapt the varying length sequences to SVMs, two possible solutions came to mind: choosing a fixed sequence length to be used in all training and testing examples and padding sequences shorter than that with 0s or padding short sequences with random values ranging from 1 to 16.

I tried both approaches, but the results of padding with 0s were much better than with random values and

therefore that's the approach that has been used in this project. Note that the possible symbols in the original sequence range from 1 to 16 and that 0 constitutes a new symbol that only appears when the sequence is shorter than the fixed length. After analyzing the collected gesture data, I chose a fixed length of 35 units as no collected gesture was longer than that.

The gestures are then represented by a 35×1 feature vector in which its components can range between 0 and 16 (discrete).

5.2 Implementation

Support Vector Machines in general find the maximum margin separator between two classes. In order find that separator, the solution to a quadratic program needs to be found (as we did in Homework # 2). For this project, instead of using the SVM implementation that I coded in HW # 2, I decided to use the most popular open source SVM library, LibSVM (Chang and Lin 2011). The main reasons why I decided to use LibSVM instead of my own implementation are that this time I am using quite a lot of data (1320 examples, having each 35 features) which could make my implementation unstable and certainly slow and that, having already coded an SVM implementation, it would be much more interesting for me to take this opportunity to learn to use a popular library that I could benefit from in the future.

5.3 Multi-class SVM

A single SVM classifier can only differentiate between two classes. However, I need much more than two classes to differentiate between all the gestures that I defined in section 2.2.

There are three main ways people use SVMs for multi-class classification: one-vs-one, one-vs-all and Directed-Acyclic Grahp SVM. I this project I have chosen to use one-vs-one multi-class classification for two reasons: it is the one that LibSVM recommends in its guide (Hsu, Chang, and Lin 2003) and a study comparing different multi-class methods concluded that one-vs-one performs very well in general (Hsu and Lin 2002).

In order to do one-vs-one multi-class classification, $k(k-1)/2$ SVM classifiers need to be trained. Then, to predict a class a voting schema is used: the predicted class is the one that achives the largest number of votes after the input data is run through all the SVM classifiers.

5.4 Selecting the SVM parameters

In order to train the SVM classifiers I followed the official LibSVM guide (Hsu, Chang, and Lin 2003) that provides very insightful tips to succeed using SVMs. In first place, the training and testing data was scaled from the 0 to 16 range to range [-1, +1] as suggested in the guide.

I also decided to use the RBF kernel which is a popular choice for SVM kernels as we studied in class. Given these choices, the only remaining parameters that need to be established are C, the cost factor, and γ, the precision of the RBF kernel.

As suggested in the LibSVM guide, C and γ were chosen using grid like search and 5-fold cross-validation.

5.5 Classification results

This section presents the results of using SVM following the method described earlier in two different scenarios. In the first scenario the objective is to classify the gestures in the testing set with the constraint that all the entries in the testing set belong to one of the 11 trained gestures. Because we are using one-vs-one multi-class classification, we need to train $11(11-1)/2 = 55$ SVM classifiers. As mentioned in the previous section, we use cross-validation to find the best values for C and γ. The values that achieved the best performance were $C = 2$ and $\gamma = 0.125$. The results of applying SVM to this scenario are presented in Table 6. Note that this scenario is the same presented in section 4.4 and therefore, Table 6 can be compared to Table 4.

HMM Classification results (within trained gestures)

#	Gesture	Tests	Correct	Errors	%
1	L	60	53	7	88.33
2	O	60	60	0	100
3	V	60	51	9	85.00
4	Z	60	53	7	88.33
5	M	60	59	1	98.33
6	G	60	59	1	98.33
7	T	60	56	4	93.33
8	And	60	56	4	93.33
9	GT	60	57	3	95
10	W	60	59	1	98.33
11	C	60	59	1	98.33
	Total	**660**	**622**	**38**	**94.2**

Table 6: SVM results. Classification within trained gestures.

As we can see in Table 6, the accuracy of this solution is *94.2%*. This was a surprise for me as I didn't expect that SVM would yield such good results given that it is not particularly well suited for sequential information. Moreover, each gesture example had a different length and all were padded artificially with 0s until the fixed length of 35 was reached, which I thought would be problematic. Although HMM still yielded a better performance (*96.81%*), the results achieved with SVM are remarkable, especially considering its straightforward implementation.

The second scenario is analogous to the one presented in section 4.5. The task in this scenario is to classify gestures in either one of the 9 trained gestures (gestures from "L" to "GT") or in the unrecognized class. In this case the gestures "W" and "C" are not trained, but examples of these gestures are introduced in the testing set together with random sequences. These are consid-

ered to be correctly classified if assigned to the *unrecognized gesture* class. Note that in this scenario there are $9 + 1 = 10$ classes and therefore $10(10-1)/2 = 45$ SVM classifiers are needed. The results of this scenario are presented in Table 7. Again, this scenario is exactly the same as the HMM one presented in section 4.5 and therefore these results can be compared to Table 5. The parameters chosen in this case were $C = 128$ and $\gamma = 0.03125$

SVM Classification results (trained and untrained)

#	Gesture	Tests	Correct	Errors	%
	Trained				
1	L	60	53	7	88.33
2	O	60	59	1	98.33
3	V	60	53	7	88.33
4	Z	60	49	11	81.67
5	M	60	57	3	95.00
6	G	60	60	0	100
7	T	60	57	3	95.00
8	And	60	55	5	91.67
9	GT	60	56	4	93.33
	Random				
	Random seq	60	59	1	98.33
	Untrained				
10	W	60	58	2	96.67
11	C	60	59	1	98.33
	Total	**720**	**675**	**45**	**93.8**

Table 7: SVM results. Trained and untrained gestures.

Again, the results achieved with SVM in this scenario (*93.8%*) are far better than expected. Under the same conditions, HMM achieved an accuracy of 96.2% which is still better.

6 Real-time gesture recognition interface

As an extra for this project, a MATLAB interface was developed to detect the previously trained gestures in real-time (Figure 7). Using this interface, the user can "draw" a gesture and whenever the mouse button is released, the program will calculate the sequence of discretized angles and use it to calculate the likelihood of the gesture belonging to any of the trained gestures. At this time this interface only works with the HMM method and not with the SVM one as the former has proven to be slightly more effective.

7 Conclusions

My objective for this project was to design a method to detect two dimensional gestures executed in a graphical user interface with either a mouse or a trackpad using the machine learning techniques learned during the semester. I was in particular very interested in using Hidden Markov Models for this application given

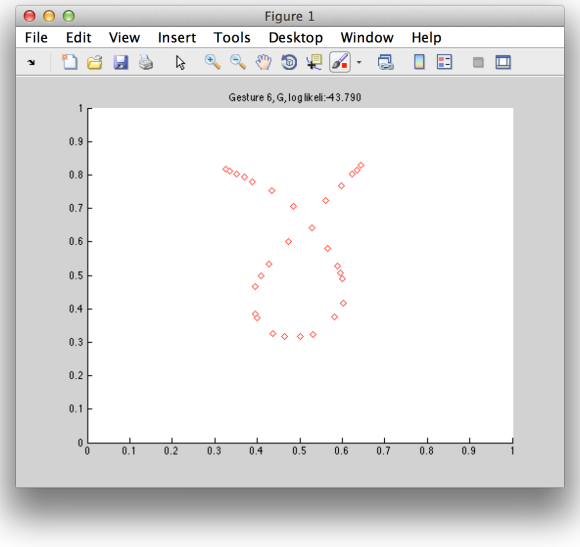

Figure 7: MATLAB Real-Time interface for gesture detection.

their popularity for classification of sequential data and because we hadn't had the chance to experiment with them in the homeworks. I believe this project has been successful in the sense that I have developed two gesture recognition systems (HMMs and SVMs) that achieve very high recognition rates and analyzed their performance using a large number of examples that I collected with a custom designed application.

There are several aspects of this project that I would like to highlight. First of all, I reach the conclusion that the gestures under consideration can be represented by a feature vector consisting on a sequence of discretized angles that is appropriate for classification. In order to assess the performance of my classifier and feature selection, I designed an experiment for which I had to develop a MATLAB application that recorded gestures and I was able to collect a total of 1320 examples of 11 types of gestures executed by two subjects, which took a considerable amount of time and effort.

Then I explored how to face the gesture classification challenge using HMMs. I provide insights on why I thought that HMMs would be a good way to solve this problem and design a system to classify the gestures using HMMs by calculating the likelihood of the observed sequence of discretized angles for each HMM model. I justified why *LRB* HMMs are better suited for this task and how to select the appropriate number of states for each model. Using this classification system, I was able to achieve classification rates of **96.81%** for a large testing set of 660 examples of the 11 trained gestures. Then, I explained how to modify this system to classify untrained gestures as well as random sequences by using *thresholds*. Using this modified system, HMMs achieved classification rates of **96.2%** when the large

training set of 720 examples was composed by examples of trained gestures as well as random sequences and examples of two non-trained gestures.

Finally, I also designed a system to classify gestures using SVMs for which the sequences of discretized angles had to be converted to fixed lengths. Much to my surprise, SVMs performed remarkably well for both the case in which only examples of trained gestures were tested (**94.2%** accuracy) and the case in which there were also examples of non-trained gestures and random sequences that had to be classified as *unrecognized* gestures (**93.8%** accuracy). Although lower, these rates are pretty close to the ones achieved by the HMM solution, that exploits the sequential nature of the recorded data. Before working on this project I did not expect SVMs to perform this well given the fact that gesture examples have different lengths and that SVMs take into account the sequential nature of the gestures.

Given that the performance of both methods is pretty high, it seems reasonable to wonder which one is better for this specific task. *HMMs* provided the best classification rates (as expected), however, they are considerably harder to train than SVMs given that there's no closed solution for the parameter estimation problem and all we can do is reach local likelihood maximums using the Baum-Welch algorithm. This is especially problematic given that the solution to the algorithm depends on the initialization conditions and it is not easy to specify reasonable initial conditions for this application other than random initialization matrices. However, even though training HMMs is not an easy task, I did not encounter many problems when using Baum-Welch and the training models using this method performed consistently well in my tests. *SVMs* on the other hand are much easier to train, converging reliably to the best solution and there are very efficient libraries for performing this task such as LibSVM. Although SVMs performed very well in my tests, there are also some problems associated to its use that are worth mentioning. An SVM classifier can only differentiate between two classes and in order to do multi-class classification, we need to use other methods such as one-vs-one classification. In my project, for 11 classes using one-vs-one classification, $11 \cdot (11-1)/2 = 55$ SVMs classifiers need to be trained and used during prediction. We can see that as soon as the number of classes grows, the number of SVM classifiers needed can become unmanageable.

I enjoyed very much working on this project although it often proved to be challenging to complete. This was my first real incursion in the world of machine learning and it was a great learning experience for me. I had the chance to apply the concepts learned in class to real data that I captured myself and I had to face challenges such as numerical instability errors, parameter selection, handling of large data sets and understand the limitations of machine learning for this type of applications.

References

Akl, A., and Valaee, S. 2010. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation. *... and Signal Processing (ICASSP)*.

Benbasat, A. Y., and Paradiso, J. A. 2002. *An Inertial Measurement Framework for Gesture Recognition and Applications*, volume 2298 of *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Chang, C. C., and Lin, C. J. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27.

Cho, S.-J. 2005. Introduction to Hidden Markov Model and Its Application. 1–36.

Dadgostar, F.; Sarrafzadeh, A.; Fan, C.; De Silva, L.; and Messom, C. 2006. Modeling and Recognition of Gesture Signals in 2D Space: A Comparison of NN and SVM Approaches. In *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, 701–704.

Elmezain, M.; Al-Hamadi, A.; Appenrodt, J.; and Michaelis, B. P. R. . I. . t. I. C. o. 2008. A Hidden Markov Model-based continuous gesture recognition system for hand motion trajectory. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*.

Hsu, C. W., and Lin, C. J. 2002. A comparison of methods for multiclass support vector machines. *Neural Networks*.

Hsu, C. W.; Chang, C. C.; and Lin, C. J. 2003. A practical guide to support vector classification.

Lee, H.-K.; Kim, J. H. P. A.; and on, M. I. I. T. 1999. An HMM-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21(10).

Liu, N.; Lovell, B. C.; and Kootsookos, P. J. 2004. Model structure selection & training algorithms for an HMM gesture recognition system. *... Recognition*.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Starner, T., and Pentland, A. C. V. . P. I. S. o. 1995. Real-time American Sign Language recognition from video using hidden Markov models. In *Computer Vision, 1995. Proceedings., International Symposium on*.

Stiefmeier, T.; Roggen, D.; and Tröster, G. 2007. Gestures are strings: efficient online gesture spotting and classification using string matching. 16.

Wobbrock, J. O.; Wilson, A. D.; and Li, Y. 2007. Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *the 20th annual ACM symposium*, 159. New York, New York, USA: ACM Press.