Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Αναγνώριση Χειρονομιών σε Οθόνη Αφής

Διπλωματικη Εργασια

του

Χρήστου Μητρόπουλου

Επιβλέπων : Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2015

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αναγνώριση Χειρονομιών σε Οθόνη Αφής

Διπλωματικη Εργασια

του

Χρήστου Μητρόπουλου

Επιβλέπων : Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Οκτωβρίου 2015.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Στέφανος Κόλλιας	Κωνσταντίνος Καρπούζης	Janne Lindqvist
Καθηγητής Ε.Μ.Π.	Ερευνητής Ε.Μ.Π.	Καθηγητής Rutgers University

Αθήνα, Οκτώβριος 2015

(Υπογραφή)

.....

Χρηστος Μητροπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Στέφανο Κόλλια, που μου επέτρεψε να εργαστώ πάνω στο θέμα που επέλεξα.

Θα ήθελα επίσης να ευχαριστήσω τον κ. Κωνσταντίνο Καρπούζη για τη διαλλακτικότητα του και τη βοήθεια του όποτε τον χρειάστηκα.

Επιπλέον θα ήθελα να ευχαριστήσω τον Professor Janne Lindqvist που με εμπιστεύτηκε με τη συγκεκριμένη εργασία και μου επέτρεψε να την συνεχίζω σαν θέμα της διπλωματικής μου. Οι γνώσεις και οι εμπειρίες που απέκτησα κατά τη διάρκεια της ανάπτυξης της διπλωματικής ήταν πολύ σημαντικές και ενδιαφέρουσες.

Θα ήθελα επίσης να ευχαριστήσω τους γονείς μου και την αδερφή μου για την στήριξη τους καθ΄ όλη τη διάρκεια των σπουδών μου.

Τέλος θα ήθελα να ευχαριστήσω τους φίλους και συνάδελφους Χατζ και Όμηρο για τις ώρες που περάσαμε μαζί διαβάζοντας, καθώς το μοναχικό διάβασμα είναι πολύ πιο κουραστικό, αλλά και όλους τους υπόλοιπους φίλους μου για την στήριξη τους και την παρέα τους εδώ και πολλά χρόνια και όχι μόνο κατά τη διάρκεια των σπουδών μου. Μάξιμε, Τζαννέ, Μπιλ, Θανάση, Χριστιάννα , Άννα-Μαρία σας ευχαριστώ.

ΠΕΡΙΛΗΨΗ

Καθημερινά ο καθένας μας για πολλούς και διαφορετικούς λόγους χρησιμοποιεί κωδικούς για να ταυτοποιηθεί προτού κάνει χρήση μιας οποιαδήποτε υπηρεσίας. Οι κλασσικοί αλφαριθμητικοί κωδικοί ενώ χρησιμοποιούνται αρκετά χρόνια τείνουν πολλές φορές να γίνουν τετριμμένοι και να θέσουν σε κίνδυνο την ασφάλεια του συστήματος που προστατεύουν. Άλλες μορφές κωδικών βασισμένες σε βιομετρικά χαρακτηριστικά έχουν προταθεί και μία από αυτές είναι η χρησιμοποίηση χειρονομιών σε οθόνες αφής, δεδομένης και της μεγάλης χρησιμοποίησης των κινητών με οθόνη αφής.

Ο σκοπός της διπλωματικής είναι η ανάπτυξη μιας εφαρμογής σε λογισμικό Android για την αναγνώριση χειρονομιών σε οθόνη αφής. Η εφαρμογή θα κάνει χρήση στοχαστικών μοντέλων για την αναγνώριση των χειρονομιών. Πιο συγκεκριμένα, θα χρησιμοποιηθεί ένα μοντέλο Μαρκοβιανής αλυσίδας, το Κρυφό Μαρκοβιανό Μοντέλο (Hidden Markov Model - HMM).

Για να δοκιμαστεί η εφαρμογή έγινε μια έρευνα σε χρήστες από τους οποίους ζητήθηκε να δημιουργήσουν δέκα όμοιες χειρονομίες οι οποίες θα χρησιμοποιηθούν για τη δημιουργία ενός Hidden Markov Model το οποίο θα είναι μοναδικό και θα αντιστοιχεί στη χειρονομία που διάλεξαν οι ίδιοι. Στη συνέχεια τους ζητήθηκε να επαναλάβουν τη χειρονομία μετά από λίγα λεπτά, καθώς και μετά από λίγες μέρες έτσι ώστε να καταγραφεί η ομοιότητα των νέων χειρονομιών με βάση τα μοντέλα που δημιουργήθηκαν.Τα δεδομένα συγκεντρώθηκαν σε μία βάση δεδομένων έτσι ώστε να μελετηθούν συνολικά.

Λέξεις Κλειδιά

Αναγνώριση χειρονομιών σε οθόνη αφής, Android, Hidden Markov Models.

Abstract

Everyone uses passwords for many and various reasons many times throughout the day, in order to verify his or her identity before using a service. Although classic alphanumerical passwords are being used for many years, they tend to be dentrimental and putting at risk the security of the system they protect. Different password forms have been introduced that are based on biometrical features and one of them is using gestures on touchscreens, considering the extended use of smartphones.

This diploma thesis aims to the development of a android app for gesture recognition. The app will use stochastic models for identifying the gestures and in particular the app wil use a Markovian chain model, the Hidden Markov Model.

In order to test the application a user study has been conducted where users have been asked to create ten identical gestures that will then be used to create a Hidden Markov Model which will be unique and representative of the user-selected gesture. All users where asked to recreate their gestures after a short and a longer time period, in order to register the similarity of the new gestures, given the created Hidden Markov Models. All data were gathered in a database.

Keywords

Gesture recognition on touch screen ,Android, Hidden Markov models.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο της διπλωματικής

Είναι γεγονός ότι για κάθε ασφαλή συναλλαγή ή δημιουργία λογαριασμού είναι απαραίτητη η δημιουργία ενός κωδικού, ο οποίος να είναι ευκολομνημόνευτος καθώς και όχι εύκολα αντιγράψιμος από άλλους. Η συνήθης πρακτική είναι η χρησιμοποίηση αλφαριθμητικών χαρακτήρων για τη δημιουργία κωδικών. Μία εναλλακτική πρακτική είναι η χρησιμοποίηση χειρονομιών πάνω σε οθόνη αφής για την αυθεντικοποίηση των χρηστών.

Σε αυτή τη διπλωματική εξετάζουμε αλγόριθμους και μεθόδους κατηγοριοποίησης χειρονομιών με βάση τα εξής χαρακτηριστικά: γωνία από το κέντρο μάζας, γωνία από το προηγούμενο σημείο, ταχύτητα στον οριζόντιο και κατακόρυφο άξονα, πίεση που ασκείται πάνω στην οθόνη αφής.

Αντιλαμβάνεται κανείς ότι η συγκεκριμένη εφαρμογή μπορεί να έχει άμεση εφαρμογή δεδομένης της μεγάλης χρησιμοποίησης smart-phones και των οθονών αφής τους, οι οποίες επιτρέπουν την εύκολη δημιουργία χειρονομιών.

Για την υλοποίηση των Hidden Markov Models σε περιβάλλον Android χρησιμοποιήθηκε η βιβλιοθήκη JaHMM η οποία παρέχει αλγόριθμους, οι οποίοι χρειάζονται για τη δημιουργία των μοντέλων, υλοποιημένους σε Java

1.2 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη στα εξής επτά κεφάλαια:

Στο κεφάλαιο 2 δίνεται το θεωρητικό υπόβαθρο των βασικών τεχνολογιών που σχετίζονται με τη διπλωματική αυτή. Πιο συγκεκριμένα, ορίζονται οι βασικές θεωρητικές έννοιες και παρουσιάζονται οι αλγόριθμοι που χρησιμοποιήθηκαν.

Στο κεφάλαιο 3 αρχικά περιγράφονται οι σχετικές με το θέμα εργασίες και στη συνέχεια δίνεται ο στόχος της συγκεκριμένης εργασίας.

Στο κεφάλαιο 4 παρουσιάζεται η ανάλυση και η σχεδίαση του συστήματος, δηλαδή η περιγραφή των υποσυστημάτων και των εφαρμογών του.

Στο κεφάλαιο 5 δίνεται η περιγραφή της υλοποίησης του συστήματος, με ανάλυση των βασικών αλγορίθμων καθώς και λεπτομέρειες σχετικά με τις πλατφόρμες και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν.

Στο κεφάλαιο 6 παρουσιάζεται ο έλεγχος καλής λειτουργίας του συστήματος με βάση τη συγκεκριμένη έρευνα χρηστών που πραγματοποιήθηκε.

Τέλος στο κεφάλαιο 7 δίνεται η συνεισφορά αυτής της διπλωματικής εργασίας καθώς και μελλοντικές επεκτάσεις.

Κεφάλαιο 2

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Μηχανική Μάθηση

Η μηχανική μάθηση (machine learning) είναι μια περιοχή της τεχνητής νοημοσύνης η οποία αφορά αλγορίθμους και μεθόδους που επιτρέπουν στους υπολογιστές να «μαθαίνουν». Με τη μηχανική μάθηση καθίσταται εφικτή η κατασκευή προσαρμόσιμων προγραμμάτων υπολογιστών τα οποία λειτουργούν με βάση την αυτοματοποιημένη ανάλυση συνόλων δεδομένων και όχι τη διαίσθηση των μηχανικών που τα προγραμμάτισαν. Η μηχανική μάθηση επικαλύπτεται σημαντικά με τη στατιστική, αφού και τα δύο πεδία μελετούν την ανάλυση δεδομένων.

Οι αλγόριθμοι μηχανικής μάθησης κατηγοριοποιούνται ανάλογα με το επιθυμητό αποτέλεσμα του αλγορίθμου. Οι συνηθέστερες κατηγορίες είναι οι εξής:

- Επιτηρούμενη μάθηση, επιβλεπόμενη μάθηση ή μάθηση με επίβλεψη όπου ο αλγόριθμος κατασκευάζει μια συνάρτηση που απεικονίζει δεδομένες εισόδους σε γνωστές, επιθυμητές εξόδους (σύνολο εκπαίδευσης), με απώτερο στόχο τη γενίκευση της συνάρτησης αυτής και για εισόδους με άγνωστη έξοδο (σύνολο ελέγχου).
- Μη επιτηρούμενη μάθηση, ανεπίβλεπτη μάθηση ή μάθηση χωρίς επίβλεψη όπου ο αλγόριθμος κατασκευάζει ένα μοντέλο για κάποιο σύνολο εισόδων χωρίς να γνωρίζει επιθυμητές εξόδους για το σύνολο εκπαίδευσης.
- Ενισχυτική μάθηση όπου ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών για μια δεδομένη παρατήρηση.

Η ανάλυση των αλγόριθμων μηχανικής μάθησης είναι ένας κλάδος της στατιστικής που ονομάζεται θεωρία μάθησης.

Η παρούσα διπλωματική ασχολείται με αλγόριθμους επιτηρούμενης μάθησης. Μηχανική μάθηση μπορεί να υπάρχει σε πληθώρα εφαρμογών, όπως Βιοτεχνολογία, Υπολογιστικά Οικονομικά και Αναγνώριση προτύπων.

2.2 Κρυφά Μαρκοβιανά Μοντέλα

Το μοντέλο που χρησιμοποιήθηκε για τη μοντελοποίηση των χειρονομιών είναι το Hidden Markov Model(HMM) (Κρυφό Μαρκοβιανό Μοντέλο), το οποίο είναι ένα

στατιστικό Μαρκοβιανό μοντέλο στο οποίο το σύστημα το οποίο μοντελοποιείται θεωρούμε ότι είναι μια μαρκοβιανή διαδικασία, δηλαδή μια διαδικασία η οποία είναι χωρίς μνήμη και αρκεί μόνο η παρούσα κατάσταση για να γίνει οποιαδήποτε πρόβλεψη για μελλοντική κατάσταση. Ακολουθεί η παρουσίαση της θεωρίας των Hidden Markov Models.

2.2.1 Μαρκοβιανή αλυσίδα

Η Μαρκοβιανή αλυσίδα είναι ένα μαθηματικό σύστημα που μεταβάλλεται από μια κατάσταση σε μία άλλη, ανάμεσα σε ένα πεπερασμένο αριθμό καταστάσεων. Είναι μια τυχαία διαδικασία που δεν διατηρεί μνήμη για τις προηγούμενες μεταβολές. Η μετάβαση από μία κατάσταση σε μια άλλη γίνεται βάσει των πιθανοτήτων μετάβασης, οι οποίες είναι διαφορετικές σε κάθε κατάσταση.

Το σύστημα επίσης παράγει μία έξοδο, ανάμεσα στις μεταβάσεις του, από ένα σύνολο συμβόλων εξόδου, η κατανομή πιθανότητας των οποίων είναι διαφορετική για κάθε κατάσταση.

2.2.2 Κρυφό Μαρκοβιανό Μοντέλο

Στα απλά Μαρκοβιανά μοντέλα η κάθε κατάσταση είναι ορατή στον παρατηρητή και επομένως οι μόνες μεταβλητές είναι οι πιθανότητες μεταβολής κατάστασης. Αντίθετα, στα Κρυφά Μαρκοβιανά Μοντέλα κάθε κατάσταση δεν είναι απευθείας ορατή στον παρατηρητή, αλλά το εξαρτώμενο από την κατάσταση αποτέλεσμα είναι ορατό. Κάθε κατάσταση έχει μία κατανομή πιθανότητας πάνω στα πιθανά αποτελέσματα και η μετάβαση από μία κατάσταση στην επόμενη γίνεται βάσει των πιθανοτήτων μετάβασης, οι οποίες είναι διαφορετικές για κάθε κατάσταση. Επομένως κάθε ακολουθία αποτελεσμάτων μπορεί να παρέχει πληροφορία για την ακολουθία των καταστάσεων.

Οι παράμετροι που αρκούν για να περιγράψουν πλήρως ένα Hidden Markov Model είναι οι εξής:

1. N, ο αριθμός των κρυφών καταστάσεων. Παρ΄ όλο που οι καταστάσεις είναι κρυφές, σε πολλές πρακτικές εφαρμογές έχουν κάποια φυσική σημασία. Στην παρούσα διπλωματική οι καταστάσεις αντιπροσωπεύουν τις διαφορετικές περιοχές της οθόνης. Εν γένει οι καταστάσεις συνδέονται ανά δύο, παρ΄ όλα αυτά οι διαφορετικοί τρόποι σύνδεσης χρησιμοποιούνται για τη μοντελοποίηση διαφορετικών φυσικών φαινομένων. Συμβολίζουμε το σύνολο των διαφορετικών καταστάσεων ως $S = \{S_1, S_2, ..., S_n\}$ και την κατάσταση τη χρονική στιγμή t ως q_t.

2. M, ο αριθμός των διαφορετικών συμβόλων παρατήρησης ανά κατάσταση, π.χ. το μέγεθος του διακριτού αλφάβητου. Τα σύμβολα παρατήρησης αντιστοιχούν στη φυσική έξοδο του συστήματος που μοντελοποιείται. Συμβολίζουμε τα διαφορετικά σύμβολα παρατήρησης ως $V = \{v_1, v_2, ..., v_n\}$

3. Την κατανομή πιθανότητας μετάβασης καταστάσεων $A = \{a_{ij}\}$, όπου

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \qquad 1 \leq i, j \leq N$$

Για την ειδική περίπτωση στην οποία κάθε κατάσταση είναι προσβάσιμη από κάθε άλλη κατάσταση με ένα βήμα, έχουμε $a_{ij} > 0$ για κάθε i, j. Για άλλους τύπους HMM, θα μπορούσαμε να έχουμε $a_{i,j} = 0$ για ένα ή περισσότερα (i, j) ζευγάρια.

4. Την κατανομή πιθανότητας των συμβόλων παρατήρησης στην κατάσταση j, $B = \{b_j(k)\}$, όπου

$$b_j(k) = P[v_k \quad at \quad t|q_t = S_j], \qquad 1 \leq j \leq N, \qquad 1 \leq i \leq M$$

5. Η κατανομή πιθανότητας για την αρχική κατάσταση $\pi = \{\pi_i\}$, όπου

$$\pi_i = P[q_1 = S_i], \qquad 1 \leq i \leq N.$$

Με δεδομένες τιμές για τα N, M, A, B και p το HMM μπορεί να χρησιμοποιηθεί σαν γεννήτρια παραγωγής ακολουθιών παρατήρησης

$$O = O_1 O_2 ... O_T$$

όπου κάθε παρατήρηση O_t είναι ένα από τα σύμβολα που ανήκουν στο V, και T είναι το πλήθος των παρατηρήσεων στην ακολουθία.

Από τα παραπάνω φαίνεται ότι για τον πλήρη προσδιορισμό ενός HMM πρέπει να καθοριστούν δύο παράμετροι του μοντέλου (N, M), τα σύμβολα παρατήρησης και τα τρία πιθανοτικά μεγέθη (A, B, π).

Για λόγους συντομίας θα χρησιμοποιείται ο συμβολισμός

$$\lambda = (A, B, \pi)$$

για να υποδείξουμε το πλήρες σύνολο παραμέτρων.

Τα Κρυφά Μαρκοβιανά Μοντέλα είναι κυρίως γνωστά για τις εφαρμογές τους στην αναγνώριση μοτίβων, όπως αναγνώριση ομιλίας, γραφής, χειρονομιών.

Για την καλύτερη κατανόηση των Κρυφών Μαρκοβιανών Μοντέλων δίνεται το παρακάτω παράδειγμα.

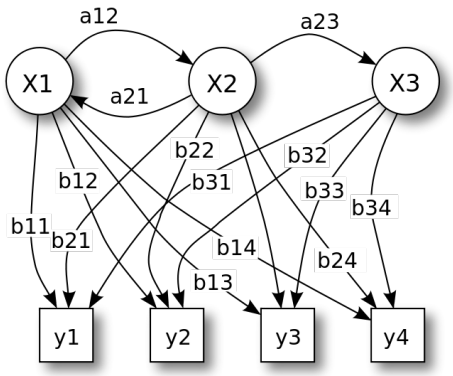
2.2.2.1 Παράδειγμα

Στη διακριτή της μορφή, μια κρυφή Μαρκοβιανή διαδικασία μπορεί να οπτικοποιηθεί σαν μια γενίκευση του προβλήματος με τις μπάλες τοποθετημένες σε δοχεία οι οποίες επιλέγονται τυχαία και στη συνέχεια επανατοποθετούνται στα δοχεία. Ας θεωρήσουμε το εξής παράδειγμα: σε ένα δωμάτιο το οποίο δεν είναι ορατό από τον εξωτερικό παρατηρητή υπάρχει ένας παντογνώστης παρατηρητής. Το δωμάτιο περιέχει δοχεία X1, X2, X3, ... καθένα από τα οποία περιέχει έναν γνωστό συνδυασμό από μπάλες, καθεμία από τις οποίες έχει μία ετικέτα y1, y2, y3,... . Ο παντογνώστης παρατηρητής διαλέγει ένα δοχείο από το δωμάτιο και τυχαία επιλέγει μία μπάλα από αυτό το δοχείο. Ο εξωτερικός παρατηρητής μπορεί να δει μόνο τη σειρά των μπαλών που διαλέγει ο παντογνώστης παρατηρητής, αλλά όχι και τα δοχεία από τα οποία διαλέγει. Ο παντογνώστης επιλέγει δοχείο με βάση μία διαδικασία. Η επιλογή του δοχείου για τη ν-οστή μπάλα εξαρτάται μόνο από έναν τυχαίο αριθμό και την επιλογή δοχείου για την (ν-1)-οστή μπάλα. Εφόσον η επιλογή δοχείου δεν εξαρτάται άμεσα από την επιλογή των δοχείων

που προηγούνται του αμέσως προηγούμενου δοχείου, αυτή η διαδικασία ονομάζεται Μαρκοβιανή διαδικασία. Μπορεί να περιγραφεί από το πάνω μέρος του Σχήματος 2.1.

Η ίδια η διαδικασία Μάρκοβ δεν μπορεί να παρατηρηθεί και μόνο η αλληλουχία των μπαλών που τραβήχτηκαν μπορεί να παρατηρηθεί. Αυτό φαίνεται από το κάτω μέρος του Σχήματος 2.1, όπου μπορούμε να παρατηρήσουμε ότι οι μπάλες y1, y2, y3, y4 μπορούν να επιλεχθούν σε κάθε κατάσταση. Ακόμα και αν ο εξωτερικός παρατηρητής ξέρει τη σύνθεση από μπάλες σε κάθε δοχείο και έχει μόλις παρατηρήσει την αλληλουχία των μπαλών που τραβήχτηκαν, π.χ. y1, y2, y3, δεν μπορεί να είναι σίγουρος για το από ποιο δοχείο τράβηξε την τρίτη μπάλα ο παντογνώστης παρατηρητής. Παρ΄ όλα αυτά ο εξωτερικός παρατηρητής μπορεί να εκτιμήσει την πιθανότητα η τρίτη μπάλα να προήλθε από καθένα από τα δοχεία.

Σχήμα 2.1: Παράδειγμα ενός Κρυφού Μαρκοβιανού Μοντέλου



2.2.3 Τύποι HMM

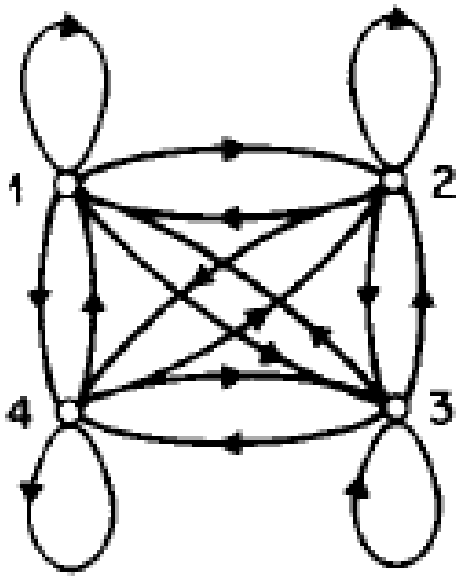
Μέχρι στιγμής έχουμε θεωρήσει μόνο την ειδική περίπτωση του εργοδικού ή πλήρους διασυνδεδεμένου HMM, στο οποίο κάθε κατάσταση θα μπορούσε να προσεγγιστεί από οποιαδήποτε άλλη κατάσταση σε ένα μόνο βήμα. Για το μοντέλο με N=4 που φαίνεται στο 2.2 ισχύει ότι όλοι οι συντελεστές μετάβασης a_{ij} είναι θετικοί.

Για μερικές εφαρμογές όμως, όπως αναγνώριση φωνής καθώς και αναγνώριση χειρονομιών, έχει βρεθεί ότι άλλοι τύποι HMM μοντελοποιούν καλύτερα το φυσικό σήμα απ΄Α ότι ένα εργοδικό μοντέλο. Ένα τέτοιο μοντέλο φαίνεται στο 2.3, το οποίο καλείται αριστερό-δεξί ή Bakis μοντέλο, γιατί έχει την ιδιότητα όσο αυξάνεται ο χρόνος, αυξάνεται επίσης και ο αριθμός της κατάστασης στην οποία βρίσκεται ή το πολύ παραμένει η ίδια.

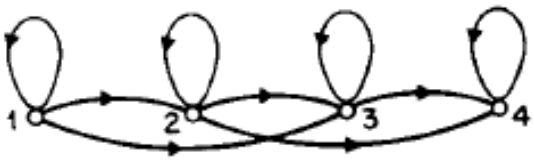
Είναι προφανές ότι τα μοντέλα αυτού του τύπου έχουν την επιθυμητή ιδιότητα να μοντελοποιούν παραστατικά σήματα τα οποία αλλάζουν με τον χρόνο. Η θεμελιώδης ιδιότητα όλων των Bakis μοντέλων είναι ότι για τους συντελεστές μετάβασης ισχύει $a_{ij} = 0$ για $i < j$. Επίσης για τις πιθανότητες των αρχικών καταστάσεων έχουμε την παρακάτω ιδιότητα

$$\pi_i = \begin{cases} 1 & \text{εάν } i = 1 \\ 0 & \text{εάν } i \neq 1 \end{cases} \tag{2.1}$$

Σχήμα 2.2: Πλήρως διασυνδεδεμένο HMM



Σχήμα 2.3: Αριστερό-δεξί HMM



καθώς η ακολουθία θα πρέπει να ξεκινάει από την κατάσταση 1. Πολλές φορές τίθενται επιπρόσθετοι περιορισμοί στους συντελεστές μετάβασης ώστε να εξασφαλιστεί ότι δεν θα συμβαίνουν μεγάλες μεταβάσεις. Ένας τέτοιος περιορισμός θα μπορούσε να ήταν $a_{ij} = 0, \quad j > i + \delta$ και για το μοντέλο του 2.3, είναι δ=2 και ο πίνακας μετάβασης είναι ο παρακάτω

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0s & 0 & a_{44} \end{bmatrix}$$

Εκτός όμως από τον διαχωρισμό των μοντέλων σε εργοδικά και δεξιά-αριστερά, υπάρχουν και πολλοί άλλοι τύποι και συνδυασμοί.

Τέλος, πρέπει να τονίσουμε πως οι περιορισμοί στα Bakis ή σε άλλα μοντέλα, δεν επηρεάζουν την διαδικασία εκτίμησης των παραμέτρων. Αυτό συμβαίνει γιατί όταν η αρχική τιμή για κάποιο στοιχείο του πίνακα μετάβασης είναι 0, το στοιχείο αυτό θα παραμείνει 0 και μετά την ολοκλήρωση της διαδικασίας της εκτίμησης των παραμέτρων.

2.2.3.1 Συνεχή HMM

Όλα όσα έχουμε αναφέρει μέχρις εδώ αφορούν HMM των οποίων οι έξοδοι είναι διακριτά σύμβολα τα οποία προέρχονται από ένα πεπερασμένο αλφάβητο και για το λόγο αυτό μπορούσαμε να χρησιμοποιήσουμε μια διακριτή κατανομή πιθανότητας για την έξοδο σε κάθε κατάσταση του μοντέλου. Το πρόβλημα με αυτή την προσέγγιση , τουλάχιστον σε μερικές εφαρμογές είναι ότι οι παρατηρήσεις (έξοδοι) είναι συνεχή σήματα. Αν και είναι δυνατόν αυτά τα σήματα να υποστούν κβαντισμό και να γίνουν διακριτά αυτό μπορεί να οδηγήσει σε σφάλμα και αλλοίωση της μορφής τους. Για το λόγο αυτό θα ήταν πολύ χρήσιμη η ύπαρξη HMM με συνεχείς κατανομές εξόδων.

Προκειμένου να χρησιμοποιηθούν συνεχείς κατανομές για τις εξόδους, πρέπει να τεθούν κάποιοι περιορισμοί στην μορφή του μοντέλου της συνάρτησης πυκνότητας πιθανότητας ώστε να εξασφαλιστεί ότι θα πετύχουμε μια αξιόπιστη εκτίμηση των παραμέτρων της. Η πιο γενική παρουσίαση της συνάρτησης πυκνότητας πιθανότητας , για την οποίαν έχει αναπτυχθεί και η διαδικασία εκτίμησης , είναι ένα μείγμα της μορφής

$$b_i(O) = \sum_{m=1}^{M} c_{jm} \Pi[O, \mu_{jm}, U_{jm}], \qquad 1 \leq j \leq N \tag{2.2}$$

όπου Ο είναι το διάνυσμα που μοντελοποιείται, c_{jm} είναι ο συντελεστής μείγματος για το μ-οστό μείγμα στην κατάσταση ϑ και Π είναι μια οποιαδήποτε λογαριθμική ή συμμετρική κατανομή (π.χ. Γκαουσιανή). Οι συντελεστές μείγματος c_{jm} ικανοποιούν τους στοχαστικούς περιορισμούς

$$\sum_{m=1}^{M} c_{jm} = 1, \qquad 1 \leq j \leq N \tag{2.3}$$

$$c_{jm} \geq 0, \qquad 1 \leq j \leq N, \quad 1 \leq m \leq M \tag{2.4}$$

έτσι ώστε να ισχύει για την κατανομή η σωστή κανονικοποίηση

$$\int_{-\infty}^{\infty} b_j(x)dx = 1, \qquad 1 \leq j \leq N \tag{2.5}$$

Η παραπάνω συνάρτηση πυκνότητας πιθανότητας μπορεί να χρησιμοποιηθεί για την προσέγγιση σχεδόν οποιασδήποτε κατανομής και άρα είναι ιδιαιτέρως πολλές εφαρμογές. Μπορεί να αποδειχτεί ότι οι τύποι για την προσέγγιση των συντελεστών των μειγμάτων της κατανομής δηλαδή για τα c_{jm},μ_{jk},U_{jk} είναι οι ακόλουθοι

$$\bar{c_{jk}} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)} \tag{2.6}$$

$$\bar{\mu_{jk}} = \frac{\sum_{t=1}^{T} \gamma_t(i,j) \cdot O_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{2.7}$$

$$\bar{U_{jk}} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (O_t - \mu_{jk})(O_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(i,j)} \tag{2.8}$$

όπου $\gamma_t(j,k)$ είναι η πιθανότητα την χρονική στιγμή t να έχουμε την κατάσταση j και το κ μείγμα να συμβάλει στη δημιουργία του O, δηλαδή

$$\gamma_t(j,k) = [\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}][\frac{c_{jk}\Pi(O_t, \mu_{jk}, U_{jk})}{\sum_{m=1}^{M} c_{jm}\Pi(O_t, \mu_{jm}, U_{jm})}] \tag{2.9}$$

Ο όρος $\gamma_t(j,k)$ είναι η γενίκευση του όρου $\gamma_t(j)$ ο οποίος χρησιμοποιείται όταν έχουμε ένα μόνο μείγμα ή όταν η κατανομή είναι διακριτή. Η διαδικασία εκτίμησης του πίνακα μετάβασης (των συντελεστών a_{ij}) είναι η ίδια όπως και στα διακριτά HMM. Η εκτίμηση για το c_{jk} είναι ο λόγος του αριθμού που το σύστημα βρίσκεται στην κατάσταση ϑ χρησιμοποιώντας το κ μείγμα προς τον συνολικό αριθμό των φορών που το σύστημα βρίσκεται στην j κατάσταση. Παρομοίως, η διαδικασία εκτίμησης για το διάνυσμα μέσης τιμής μ_{jk} χρησιμοποιεί ως βάρη στον αριθμητή της 2.6 τις παρατηρήσεις O. Παρόμοια είναι η ερμηνεία και για την διαδικασία εκτίμησης του πίνακα ετεροσυσχέτισης U_{jk}.

2.2.4 Τα Τρία Βασικά Προβλήματα των HMM

Για την πρακτική χρησιμοποίηση των Hidden Markov Models, ϑα πρέπει να λυθούν τα εξής τρία βασικά προβλήματα:

Πρόβλημα 1: Δεδομένης μιας ακολουθίας παρατηρήσεων, $O = O_1O_2...O_T$, και του μοντέλου λ, πώς υπολογίζεται αποδοτικά η $P(O|\lambda)$ πιθανότητα να παρατηρηθεί η συγκεκριμένη ακολουθία, δεδομένου του μοντέλου λ;

Πρόβλημα 2: Δεδομένης μιας ακολουθίας παρατηρήσεων, $O = O_1O_2...O_T$, και του μοντέλου λ, πώς διαλέγουμε μια ακολουθία καταστάσεων $Q = q_1q_2...q_T$ που να αντιστοιχεί με κάποιο βέλτιστο τρόπο στις παρατηρήσεις;

Πρόβλημα 3: Πώς διορθώνουμε τις παραμέτρους $\lambda = (A, B, \pi)$ του μοντέλου έτσι ώστε να μεγιστοποιηθεί η $P(O|\lambda)$;

2.2.4.1 Εκτίμηση Πιθανότητας Ακολουθίας Εξόδου - Πρόβλημα 1

Το πρόβλημα 1 είναι το πρόβλημα της εκτίμησης, δηλαδή δοσμένου ενός μοντέλου και μιας ακολουθίας εξόδου, πώς υπολογίζεται η πιθανότητα να έχει παραχθεί η δεδομένη ακολουθία από το συγκεκριμένο μοντέλο. Το πρόβλημα μπορεί επίσης να θεωρηθεί και ως πρόβλημα ταιριάσματος, του πόσο ταιριάζει η συγκεκριμένη ακολουθία με το μοντέλο. Η τελευταία προσέγγιση είναι ιδιαιτέρως χρήσιμη όταν έχουμε μια δοσμένη ακολουθία και πολλά μοντέλα από τα οποία μπορεί αυτή να έχει προέλθει. Στην παρούσα διπλωματική το πόσο θα ταιριάζει μία ακολουθία εξόδου στο δεδομένο μοντέλο θα χρησιμοποιηθεί για την επιτυχή ή όχι ταυτοποίηση του χρήστη.

Η λύση στο Πρόβλημα 1 θα προέλθει κάνοντας χρήση του Forward-Backward αλγόριθμου και είναι η εξής:

Λύση Προβλήματος 1:

Θέλουμε να υπολογίσουμε την πιθανότητα της παρατηρούμενης ακολουθίας, $O = O_1O_2...O_T$, δεδομένου του μοντέλου λ, δηλαδή την $P(O/\lambda)$. Ο πιο άμεσος τρόπος για να γίνει αυτό είναι η εξαντλητική μέθοδος, η οποία είναι η απαρίθμηση όλων των πιθανών ακολουθιών καταστάσεων μεγέθους T(το πλήθος των παρατηρήσεων). Θεωρούμε μια τέτοια σταθερή ακολουθία καταστάσεων

$$Q = q_1q_2...q_T \tag{2.10}$$

όπου q_T είναι η αρχική κατάσταση. Η πιθανότητα της ακολουθίας παρατήρησης O για την ακολουθία της 2.10 είναι

$$P(O|Q,\lambda) = \prod_{t=1}^{T} \quad P(O_t|q_t,\lambda) \tag{2.11}$$

όπου έχουμε θεωρήσει στατιστική ανεξαρτησία μεταξύ των παρατηρήσεων,οπότε έχουμε

$$P(O|Q,\lambda) = b_{q_1}(O_1) \quad . \quad b_{q_2}(O_2) \quad ... \quad b_{q_T}(O_T) \tag{2.12}$$

Η πιθανότητα μιας τέτοιας ακολουθίας καταστάσεων Q μπορεί να γραφτεί ως

$$P(Q,\lambda) = \pi_{q_1}a_{q_1q_2}a_{q_2q_3}...a_{q_{t-1}q_t} \tag{2.13}$$

Η κοινή πιθανότητα των O και Q, δηλαδή η πιθανότητα να συμβούν ταυτόχρονα τα O και Q, είναι το γινόμενο των δυο παραπάνω όρων, δηλαδή,

$$P(O,Q|\lambda) = P(O|Q,\lambda)P(Q,\lambda) \tag{2.14}$$

Η πιθανότητα να έχουμε την ακολουθία εξόδου O με δεδομένο το μοντέλο, υπολογίζεται ως άθροισμα των παραπάνω δεσμευμένων πιθανοτήτων για όλες τις πιθανές ακολουθίες q και είναι

$$\begin{aligned} P(O|\lambda) &= \sum_{allQ} P(O|Q,\lambda)P(Q,\lambda) \\ &= \sum_{q_1q_2...q_T} \pi_{q_1}b_{q_1}(O_1)a_{q_1q_2}b_{q_2}(O_2)...a_{q_{t-1}q_t}b_{q_T}(Q_T) \end{aligned} \tag{2.15}$$

Συμπεραίνουμε εύκολα ότι η παραπάνω λύση έχει τεράστιο υπολογιστικό κόστος, εφόσον ο αλγόριθμος είναι πολυπλοκότητας $O(2TN^T)$ και ακόμα και για μια απλή περίπτωση με N=5 καταστάσεις και T=100 παρατηρήσεις, οι υπολογισμοί που απαιτούνται είναι της τάξης του $2 \times 100 \times 5^100 \approx 10^{72}$. Μια πιο αποδοτική λύση είναι η εμπρόσ-πίσω διαδικασία (forward-backward και περιγράφεται στη συνέχεια.

Forward-Backward διαδικασία: Θεωρούμε την forward μεταβλητή

$$a_t(i) = P(O_1 O_2 ... O_t, q_t = S_i | \lambda) \tag{2.16}$$

δηλαδή, την πιθανότητα της μερικής ακολουθίας παρατήρησης, $O_1, O_2 ... O_t$, (μέχρι χρόνο t) και κατάσταση S_i τη χρονική στιγμή t δεδομένου του μοντέλου λ. Μπορούμε να λύσουμε ως προς $a_t(i)$ επαγωγικά ως εξής:

1. Αρχικοποίηση:

$$a_1(i) = \pi b_i(O_1), \qquad 1 \leq i \leq N \tag{2.17}$$

2. Επαγωγή:

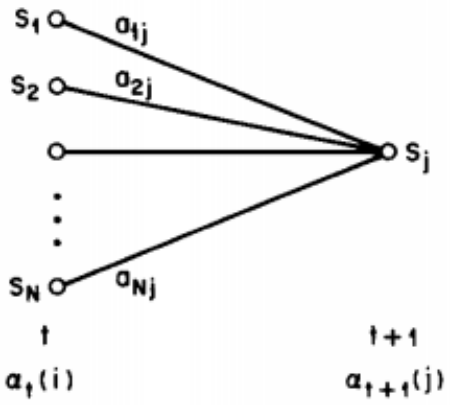
$$a_{t+1}(j) = [\sum_{i=1}^{N} a_t(i) a_{ij}] b_j(O_{t+1}), \qquad 1 \leq t \leq T-1 \quad 1 \leq j \leq N. \tag{2.18}$$

3. Τερματισμός:

$$P(O|\lambda) = \sum_{i=1}^{N} a_t(i). \tag{2.19}$$

Το βήμα 1 αρχικοποιεί την προς τα εμπρός πιθανότητα ως την δεσμευμένη πιθανότητα της κατάστασης S_i όταν η πρώτη παρατήρηση είναι η O_1. Το βήμα της επαγωγής αποτελεί και τον πυρήνα της διαδικασίας και απεικονίζεται στην εικόνα 2.4. Η συγκεκριμένη εικόνα δείχνει πως μπορούμε να οδηγηθούμε στην κατάσταση S_j τη χρονική στιγμή $t+1$ από N δυνατές καταστάσεις, $S_i, \quad 1 \leq i \leq N$ στη χρονική στιγμή t. Εφόσον $a_t(i)$ είναι η πιθανότητα της τομής των ενδεχομένων να παρατηρηθεί η ακολουθία $O_1 O_2 ... O_t$ και η κατάσταση τη χρονική στιγμή t να είναι S_i, τότε το γινόμενο $a_t(i) a_{ij}$ είναι η πιθανότητα της τομής των ενδεχομένων να παρατηρηθεί η ακολουθία $O_1 O_2 ... O_t$ και να βρεθούμε στην κατάσταση S_j τη χρονική στιγμή $t+1$ μέσω της κατάστασης S_i τη χρονική στιγμή t. Αθροίζοντας το γινόμενο πάνω σε όλες τις N πιθανές καταστάσεις $S_i, \quad 1 \leq i \leq N$ τη στιγμή t παίρνουμε σαν αποτέλεσμα την πιθανότητα την χρονική στιγμή $t+1$, να προκύψει η κατάσταση S_j με δεδομένη τη μέχρι εκείνη τη στιγμή ακολουθία εξόδου. Αφού υπολογιστεί το S_j, το $a_{t+1}(j)$ υπολογίζεται εύκολα με πολλαπλασιασμό του αθροίσματος που έχει προκύψει με την πιθανότητα στην κατάσταση j να έχουμε την έξοδο O_{t+1}, δηλαδή $b_j(O_{t+1})$. Ο παραπάνω υπολογισμός γίνεται για όλες τις καταστάσεις $S_j, \quad 1 \leq j \leq N$, για δεδομένο t και εν συνεχεία επαναλαμβάνετα για $t = 1, 2, ...T-1$. Τέλος το τρίτο βήμα μας δίνει την επιθυμητή τιμή της $P(O|\lambda)$ σαν το άθροισμα όλων των τελικών προς τα εμπρός μεταβλητών $a_T(i)$.

Σχήμα 2.4: Απεικόνιση της σειράς των διαδικασιών που απαιτούνται για τον υπολογισμό της προς τα εμπρός μεταβλητής $a_{t+1}(j)$



Η αποδοτικότητα του συγκεκριμένου αλγορίθμου είναι σαφώς καλύτερη από του προηγούμενου, εφόσον οι υπολογισμοί που χρειάζονται είναι της τάξεως του $O(N2T)$, γεγονός που μεταφράζεται σε 3000 υπολογισμούς (αντί για 10^{72}) για το προαναφερθέν μέγεθος εισόδου (N=5 και T=100). Το στοιχείο που αυξάνει την αποδοτικότητα του αλγορίθμου είναι ότι αφού υπάρχουν μόνο N καταστάσεις κάθε χρονική στιγμή, όλες οι δυνατές ακολουθίες θα προκύπτουν από τις ίδιες N καταστάσεις κάθε χρονική στιγμή. Έτσι τη χρονική στιγμή $t = 1$ υπολογίζουμε τις τιμές $a_t(i), \quad 1 \leq i \leq N$. Τις χρονικές στιγμές $t = 2, 3, ..., T$ χρειάζεται να υπολογίσουμε μόνο τις τιμές $a_t(j) \quad 1 \leq j \leq N$ και ο κάθε υπολογισμός περιλαμβάνει μόνο N προηγούμενες τιμές του $a_{t-1}(i)$ γιατί καθεμιά από τις N καταστάσεις είναι η επόμενη κάποιας από τις N, ίδιες καταστάσεις της προηγούμενης χρονικής στιγμής.

Με αντίστοιχο τρόπο ορίζεται και η προς τα πίσω (Backward) διαδικασία και η προς τα πίσω μεταβλητή $\beta_t(i)$. Η προς τα πίσω διαδικασία βέβαια δεν είναι απαραίτητη για την επίλυση του προβλήματος 1, αλλά χρησιμοποιείται για την επίλυση του προβλήματος εκπαίδευσης του μοντέλου.

2.2.4.2 Βέλτιστη Ακολουθία καταστάσεων-Πρόβλημα 2

Το πρόβλημα 2 έχει να κάνει με την αποκάλυψη του κρυφού κομματιού του μοντέλου (π.χ. να βρεθεί η σωστή ακολουθία καταστάσεων). Είναι προφανές ότι πλην ορισμένων εκφυλισμένων μοντέλων δεν υπάρχει "σωστή" ακολουθία. Αντί αυτού επιχειρείται η βελτιστοποίηση κάποιου κριτηρίου ώστε να δοθεί η καλύτερη δυνατή λύση. Εφόσον υπάρχουν πολλα κριτήρια βελτιστοποίησης, κάθε φορά επιλέγεται το κατάλληλο με βάση την εφαρμογή. Στη συγκεκριμένη διπλωματική εργασία δεν θα μας απασχολήσει η παραγωγή μιας βέλτιστης ακολουθίας καταστάσεων, επομένως δεν θα αναλυθούν περαιτέρω αλγόριθμοι βελτιστοποίησης.

2.2.4.3 Εκπαίδευση HMM - Πρόβλημα 3

Το πρόβλημα 3 είναι το πρόβλημα στο οποίο προσπαθούμε να βελτιστοποιήσουμε τις παραμέτρους του μοντέλου, έτσι ώστε να περιγράφουν με βέλτιστο τρόπο πώς προκύπτει μια δεδομένη ακολουθία εξόδου. Η ακολουθία εξόδου που χρησιμοποιείται για να προσαρμόσει τις παραμέτρους καλείται ακολουθία εκπαίδευσης, καθώς με αυτή γίνεται η εκμάθηση του μοντέλου, δηλαδή η βελτιστοποίηση των παραμέτρων του. Το πρόβλημα αυτό είναι ιδιαίτερα σημαντικό στις περισσότερες εφαρμογές των HMM, καθώς χρειάζεται οι παράμετροι του μοντέλου να προσαρμοστούν κατά βέλτιστο τρόπο στην ακολουθία εξόδου, έτσι ώστε να μοντελοποιούν το φυσικό φαινόμενο που περιγράφεται όσο το δυνατόν καλύτερα.

Στην πραγματικότητα, δοσμένης μιας πεπερασμένης ακολουθίας εξόδων σαν δεδομένα εκπαίδευσης, δεν υπάρχει βέλτιστος τρόπος εκτίμησης των παραμέτρων του μοντέλου. Μπορούμε παρ΄ όλα αυτά να διαλέξουμε λ=(A,B,π) έτσι ώστε η $P(O|\lambda)$ να μεγιστοποιείται τοπικά με χρήση κάποιας επαναληπτικής μεθόδου, όπως ο αλγόριθμος Baum-Welch ο οποίος περιγράφεται παρακάτω:

Λύση Προβλήματος 3:

Για να περιγραφεί η μέθοδος επανεκτίμησης (επαναληπτική ενημέρωση και βελτίωση) των παραμέτρων του HMM ορίζουμε την πιθανότητα $\xi_t(i,j)$, να είμαστε στην κατάσταση s_i τη χρονική στιγμή t και στην s_j τη χρονική στιγμή $t+1$, δεδομένου του μοντέλου και της ακολουθίας εξόδου, δηλαδή

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{2.20}$$

η παραπάνω πιθανότητα ορίζεται και συναρτήσει των παραμέτρων $\alpha_t(i)$ και$\beta_t(j)$ του Forward-Backward αλγόριθμου.

$$\xi_t(i,j) = \frac{\alpha_t(i)\alpha_{ij}b_j(O_{t+1}\beta_{t+1}(j)}{P(O|\lambda)} \tag{2.21}$$

Ορίζοντας την πιθανότητα $\gamma_t(i)$ ως την πιθανότητα να είμαστε στην κατάσταση S_i τον χρόνο t για δεδομένο μοντέλο και ακολουθία παρατηρήσεων O. Αν αθροίσουμε για όλα τα j, τις τιμές της $\xi_t(i,j)$ λαμβάνουμε την $\gamma_t(i)$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{2.22}$$

Αν αθροίσουμε την $\gamma_t(i)$ για όλες τις χρονικές στιγμές t παίρνουμε τον αναμενόμενο αριθμό των φορών κατά τις οποίες το σύστημα φτάνει στην κατάσταση S_i στη διάρκεια του χρόνου των παρατηρήσεων. Από τα παραπάνω προκύπτει ότι το άθροισμα της χρονικής στιγμής $\xi_t(i,j)$ στο χρόνο είναι ο αναμενόμενος αριθμός των μεταφορών από την κατάσταση S_i στην S_j

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{ αναμενόμενος αριθμός μεταβάσεων από } \quad S_i \tag{2.23}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{ αναμενόμενος αριθμός μεταφορών από } \quad S_i \text{ στην } \quad S_j \tag{2.24}$$

Χρησιμοποιώντας τις παραπάνω σχέσεις έχουμε μια μέθοδο επανεκτίμησης των παραμέτρων λ=(A,B,π) του HMM. Οι εξισώσεις για τα A,B και π είναι:

$$\bar{\pi_i} = \text{αναμενόμενη συχνότητα της κατάστασης } S_i \text{ τη χρονική στιγμή } (t=1) = \gamma_1(i) \tag{2.25}$$

$$\begin{aligned} \bar{a_{ij}} &= \frac{\text{αναμενόμενος αριθμός μεταφορών από } \quad S_i \quad \text{στην} \quad S_j}{\text{αναμενόμενος αριθμός μεταβάσεων από } \quad S_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned} \tag{2.26}$$

$$\begin{aligned} \bar{b_j(k)} &= \frac{\text{αναμενόμενος αριθμός στην κατάσταση } j \text{ και παρατηρούμενο σύμβολο } \quad v_k}{\text{αναμενόμενος αριθμός στην κατάσταση } j} \\ &= \frac{\sum_{t=1|s.t.O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned} \tag{2.27}$$

Αν ορίσουμε το τρέχον μοντέλο ως λ=(A,B,π) και το χρησιμοποιήσουμε για τον υπολογισμό των δεξιών μελών των εξισώσεων 2.25-2.27 και ορίσουμε αναθεωρημένο μοντέλο $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ όπως προσδιορίζεται από τις σχέσεις 2.25-2.27, τότε έχει αποδειχθεί ότι το αναθεωρημένο μοντέλο θα είναι καλύτερο ή το ίδιο καλό σε σχέση με το προηγούμενο μοντέλο, δηλαδή $P(O|\bar{\lambda}) > P(O|\lambda)$.

Οι παράμετροι του μοντέλου επανεκτιμούνται μέχρι οι δύο πιθανότητες να μοιάζουν αρκετά με βάση ένα επίπεδο ακρίβειας ή μέχρι να ολοκληρωθεί συγκεκριμένος αριθμός επαναλήψεων.

2.2.4.4 Αρχικοποίηση HMM

Για την εκπαίδευση των HMM απαιτείται επίσης η σωστή αρχικοποίηση τους. Μια συνηθισμένη πρακτική που ακολουθείται είναι η αρχικοποίηση των HMM με χρήση του αλγόριθμου K-means. Ο συγκεκριμένος αλγόριθμος λειτουργεί ως εξής:
Δεδομένου ενός συνόλου από k κεντροειδή στοιχεία (στοιχεία αντιπρόσωποι) $m_1, m_2, ..., m_k$, ο αλγόριθμος προχωρά εναλλασσόμενος μεταξύ των δύο παρακάτω βημάτων

1. Βήμα ανάθεσης: Αναθέτουμε κάθε παρατήρηση σε μια συστάδα της οποίας το κεντροειδές στοιχείο είναι πιο κοντά. Η απόσταση υπολογίζεται με βάση την Ευκλείδεια απόσταση δύο παρατηρήσεων για δεδομένη διάσταση χώρου.

   $$S_i^{(t)} = \{x_p : ||x_p - m_i^{(t)}||^2 \leq ||x_p - m_j^{(t)}||^2 \forall j, \quad 1 \leq j \leq k\}$$

   όπου κάθε x_p τοποθετείται σε ακριβώς μία συστάδα $S^{(t)}$, ακόμα και εάν μπορεί να τοποθετηθεί σε δύο η παραπάνω συστάδες.

2. Βήμα ανανέωσης: Υπολογίζουμε τα νέα κεντροειδή στοιχεία στις συστάδες που έχουν διαμορφωθεί στο προηγούμενο βήμα ως εξής:

   $$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Ο αλγόριθμος συγκλίνει όταν κατά το βήμα ανάθεσης δεν παρατηρούμε καμία αλλαγή. Αφού και τα δύο βήματα βελτιστοποιούν την αντικειμενική συνάρτηση αθροίσματος τετραγώνων εντός των συστάδων και εφόσον υπάρχουν πεπερασμένοι τρόποι τέτοιων διαχωρισμών, ο αλγόριθμος πρέπει να συγκλίνει σε ένα τοπικό μέγιστο. Ο αλγόριθμος δεν εγγυάται την εύρεση ενός ολικού μέγιστου.

2.3 Πίνακες Σύγχυσης

Στο πεδίο της μηχανικής μάθησης, ο πίνακας σύγχυσης (Confusion Matrix), είναι μία συγκεκριμένη τετραγωνική διάταξη που μας επιτρέπει να οπτικοποιήσουμε την επίδοση ενός αλγορίθμου, τυπικά ενός αλγορίθμου επιτηρούμενης μάθησης. Κάθε στήλη του πίνακα αναπαριστά ένα στιγμιότυπο της προβλεπόμενης κλάσης, ενώ κάθε γραμμή αναπαριστά στιγμιότυπα των πραγματικών κλάσεων. Το όνομα προκύπτει από το γεγονός ότι μπορούμε εύκολα να καταλάβουμε εάν το σύστημα παρερμηνεύει κάποιες κλάσεις και τις μπερδεύει με άλλες. Κάθε στιγμιότυπο θα ανήκει σε μία από τις εξής κλάσεις, ανάλογα με το αν έχει κατηγοριοποιηθεί σωστά ή όχι:

- Ορθώς Θετικά - True Positive (TP) : Τα αποτελέσματα τα οποία καταχωρήθηκαν στην υπό εξέταση κλάση και ανήκουν όντως σε αυτή.
- Ορθώς Αρνητικά - True Negative (TN) : Τα αποτελέσματα που δεν καταχωρήθηκαν στην υπό εξέταση κλάση και δεν ανήκουν όντως σε αυτή.
- Λανθασμένα Θετικά - False Positive (FP) :Τα αποτελέσματα που καταχωρήθηκαν στην υπό εξέταση κλάση και δεν ανήκουν στην πραγματικότητα σε αυτή.
- Λανθασμένα Αρνητικά - False Negative (FN) :Τα αποτελέσματα που δεν καταχωρήθηκαν στην υπό εξέταση κλάση ενώ ανήκουν σε αυτή.

Από τον παραπάνω πίνακα μπορούμε να αντλήσουμε σημαντικά συμπεράσματα για το σύστημα από τους εξής δείκτες:

- Ευαισθησία (sensitivity) του συστήματος, δηλαδή το ποσοστό των θετικών παρατηρήσεων οι οποίες αναγνωρίζονται ως θετικές (π.χ. το ποσοστό των άρρωστων ανθρώπων που αναγνωρίζονται επιτυχώς ότι πάσχουν από τη συγκεκριμένη ασθένεια).

$$sensitivity = \frac{TP}{TP + FN} \tag{2.28}$$

- Ειδικότητα (specificity) του συστήματος, δηλαδή το ποσοστό των αρνητικών παρατηρήσεων που αναγνωρίζονται ως αρνητικές.(π.χ. το ποσοστό των υγιών ανθρώπων που αναγνωρίζονται επιτυχώς ότι είναι υγιείς).

$$specificity = \frac{TN}{FP + TN} \tag{2.29}$$

- Θετική ικανότητα πρόβλεψης (positive predictive value) του συστήματος, δηλαδή το ποσοστό των θετικών αποτελεσμάτων που έχουν αναγνωριστεί στη σωστή κλάση τους επί του συνόλου των θετικών αποτελεσμάτων

$$positive \quad predictive \quad value = \frac{TP}{TP + FP} \tag{2.30}$$

- Αρνητική ικανότητα πρόβλεψης(negative predictive value) του συστήματος, δηλαδή το ποσοστό των αρνητικών αποτελεσμάτων που ορθώς δεν έχουν καταχωρηθεί στην υπό εξέταση κλάση επί του συνόλου των αρνητικών αποτελεσμάτων

$$negative \quad predictive \quad value = \frac{TN}{TN + FN} \tag{2.31}$$

- Ακρίβεια (accuracy) του συστήματος, δηλαδή το ποσοστό των σωστών καταχωρήσεων επί του συνόλου των καταχωρήσεων

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.32}$$

2.4 Κωδικοί Πρόσβασης

Η ισχύς κωδικού πρόσβασης είναι ένα μέτρο για την αντοχή ενός κωδικού σε επιθέσεις ωμής βίας και εικασίας. Στην συνηθισμένη της μορφή, εκτιμά τον αριθμό των προσπαθειών, κατά μέσο όρο, που θα χρειαζόταν ένας κακόβουλος χρήστης για να μαντέψει σωστά τον κωδικό. Η ισχύς του κωδικού είναι συνάρτηση του μήκους, της πολυπλοκότητας και του πόσο απρόβλεπτος είναι ο κωδικός.

Η χρησιμοποίηση ισχυρών κωδικών μειώνει τον γενικό κίνδυνο να παραβιαστεί ένα σύστημα, χωρίς αυτό να σημαίνει ότι δεν χρειάζονται και άλλα μέτρα ασφάλειας.

2.4.1 Δημιουργία Κωδικού Πρόσβασης

Οι κωδικοί δημιουργούνται είτε αυτόματα (χρησιμοποιώντας εξοπλισμό τυχαιοποίησης), είτε, τις περισσότερες φορές, από τον ίδιο το χρήστη. Ενώ γίνεται να υπολογιστεί με ακρίβεια η ισχύς ενός τυχαία επιλεγμένου κωδικού απέναντι σε μια επίθεση ωμής βίας, το να υπολογιστεί η ισχύς ενός κωδικού που έχει παραχθεί από άνθρωπο είναι μια πρόκληση.

Τυπικά, ζητείται από τους χρήστες να διαλέξουν ένα κωδικό, μερικές φορές καθοδηγούμενοι από οδηγίες ή περιορισμένοι από κανόνες, όταν θέλουν να δημιουργήσουν ένα νέο κωδικό για ένα σύστημα υπολογιστή ή μία ιστοσελίδα. Μόνο προσεγγιστικές εκτιμήσεις μπορούν να γίνουν για τη δύναμη αυτών των κωδικών καθώς οι άνθρωποι τείνουν να ακολουθούν μοτίβα σε ανάλογες διαδικασίες και αυτά τα μοτίβα μπορούν συνήθως να βοηθήσουν τον επιτιθέμενο κακόβουλο χρήστη. Επιπλέον, λίστες συνηθισμένων κωδικών είναι διαθέσιμες για να χρησιμοποιηθούν από προγράμματα εύρεσης κωδικών. Αυτές οι λίστες περιέχουν αρκετά λεξικά για διάφορες γλώσσες, κωδικούς από παραβιασμένες βάσεις δεδομένων κοινωνικών δικτύων, σε απλή ή κατακερματισμένη μορφή. Έρευνες δείχνουν ότι περίπου το 40% των κωδικών που χρησιμοποιούνται

σε συστήματα υπολογιστών με πολλούς χρήστες μπορούν να βρεθούν με τα προαναφερθέντα προγράμματα, με το ποσοστό να ανεβαίνει όταν ο επιτιθέμενος χρησιμοποιεί περισσότερες προσωπικές πληροφορίες για τον χρήστη στον οποίο επιτίθεται.

2.4.2 Επικύρωση Κωδικού Πρόσβασης

Τα συστήματα που χρησιμοποιούν κωδικούς για αυθεντικοποίηση θα πρέπει να έχουν κάποιο τρόπο ώστε να ελέγχουν κάθε κωδικό που χρησιμοποιείται για να επιτραπεί η πρόσβαση στο σύστημα. Ένα ασφαλές σύστημα θα πρέπει σε κάθε περίπτωση να έχει μηδενικές λανθασμένες αυθεντικοποιήσεις χρηστών, δηλαδή δεν θα πρέπει να επιτρέπεται η πρόσβαση ενός χρήστη σε λογαριασμό στον οποίο δεν θα έπρεπε να έχει πρόσβαση, ενώ στην ιδανική περίπτωση θα πρέπει να έχει και μηδενικές λανθασμένες απορρίψεις χρηστών, δηλαδή δεν θα πρέπει να απαγορεύεται η είσοδος σε ένα χρήστης ο οποίος θα πρέπει να έχει πρόσβαση στο συγκεκριμένο λογαριασμό.

Εάν οι έγκυροι κωδικοί αποθηκεύονται απλά σε μία βάση δεδομένων, ένας επιτιθέμενος που αποκτά πρόσβαση στο σύστημα έχει αυτόματα πρόσβαση στους λογαριασμούς όλων των χρηστών. Η λύση που προτείνεται και υλοποιείται συνήθως είναι η αποθήκευση του κωδικού σε κρυπτογραφημένη και κατακερματισμένη μορφή.

2.4.3 Εντροπία σαν μέτρο της ισχύος ενός κωδικού

Είναι συνήθης πρακτική στην βιομηχανία των υπολογιστών να προσδιορίζεται η ισχύς ενός κωδικού στη μορφή της εντροπίας πληροφορίας, η οποία μετριέται σε bits. Αντί του αριθμού προσπαθειών που χρειάζονται για να βρεθεί ο κωδικός, χρησιμοποιούμε τον λογάριθμο με βάση το 2 αυτού του αριθμού, ο οποίος είναι ο αριθμός των "bits εντροπίας" σε ένα κωδικό. Ένας κωδικός με 42 bits ισχύος υπολογισμένα με τον παραπάνω τρόπο είναι το ίδιο δυνατός με μια συμβολοσειρά 42 bits επιλεγμένων τυχαία από ένα δίκαιο νόμισμα. Από μια άλλη οπτική, ένας κωδικός με 42 bits ισχύος θα χρειαζόταν 2^{42} προσπάθειες για να εξαντληθούν όλα τα πιθανά ενδεχόμενα σε μια επίθεση ωμής βίας. Στη μέση περίπτωση ένα επιτιθέμενος θα χρειαστεί τις μισές προσπάθειες για να βρει το σωστό κωδικό.

Κεφάλαιο 3

ΕΡΓΑΣΙΕΣ ΠΑΝΩ ΣΤΟ ΑΝΤΙΚΕΙΜΕΝΟ

Στο κεφάλαιο αυτό γίνεται μια περιγραφή εργασιών που έχουν ασχοληθεί με αναγνώριση χειρονομιών ή γραπτού λόγου κάνοντας χρήση Hidden Markov Models

3.1 Σχετικές εργασίες

Η αναγνώριση χειρονομιών σε οθόνη αφής χρησιμοποιώντας Hidden Markov Models παρουσιάζει ομοιότητες με εργασίες οι οποίες ασχολούνται με αναγνώριση γραπτού λόγου ή υπογραφών ειδικότερα. Στη συνέχεια αναφέρονται εργασίες οι οποίες είναι σχετικές με το θέμα τος παρούσας διπλωματικής.

3.1.1 Σύστημα αναγνώρισης υπογραφών με διακριτά HMM

Το συγκεκριμένο σύστημα [1] αναπτύχθηκε με σκοπό την αναγνώριση υπογραφών. Αποτελείται από τα εξής μέρη:

- Τμηματοποίηση υπογραφών και εξαγωγή χαρακτηριστικών
- Διαδικασία εκπαίδευσης των HMM
- Διαδικασία αυθεντικοποίησης υπογραφών
- Πειράματα

3.1.1.1 Τμηματοποίηση Υπογραφών - Εξαγωγή χαρακτηριστικών

Όλες οι υπογραφές ελήφθησαν σε ένα άσπρο χαρτί Α4 χρησιμοποιώντας μπλε ή μαύρο μελάνι. Μετά το σκανάρισμα των σελίδων, όλες οι υπογραφές ψηφιοποιήθηκαν και αφού επικολλήθηκαν σε μια τετράγωνη περιοχή, σώθηκαν σε ξεχωριστά αρχεία η καθεμία.

Η οριζόντια τμηματοποίηση της περιοχής κάθε υπογραφής έγινε χωρίζοντας την περιοχή σε κελιά. Όσο περισσότερα κελιά χρησιμοποιηθούν τόσο πιο αποδοτικά θα εξαχθούν τα χαρακτηριστικά. Ο αριθμός των κελιών που χρησιμοποιούνται είναι 4,10,25.

Η κατακόρυφη τμηματοποίηση γίνεται ανάλογα με τα pixels κάθε περιοχής και πιο συγκεκριμένα κάθε κελί αποτελείται από 16 pixels. Ο αριθμός των κατακόρυφων κελιών εξαρτάται προφανώς από το μήκος της υπογραφής.

Τα χαρακτηριστικά που εξάγονται είναι τα εξής

- Διαμέτρημα: ενσωματώνει το ύψος και το πλάτος της υπογραφής.
- Αναλογία: η γεωμετρική κανονικότητα εκφράζεται μέσα από το ποσοστό που κατέχει η υπογραφή σε κάθε κελί.
- Ελεύθερος χώρος: εκφράζεται η γεωμετρική συμπεριφορά της υπογραφής σε κάθε κελί.
- Γωνία με τον οριζόντιο άξονα: εκφράζεται η απόκλιση από τον οριζόντιο άξονα κατά τη γραφή.

Εφόσον το σύστημα χρησιμοποιεί διακριτά HMM, είναι απαραίτητη η κβαντοποίηση του κάθε διανύσματος χαρακτηριστικών, δηλαδή η δημιουργία ενός codebook και η αντιστοίχιση κάθε διανύσματος με ένα από τα στοιχεία του codebook.

3.1.1.2 Διαδικασία εκπαίδευσης των HMM

Ο σκοπός αυτής της φάσης είναι η δημιουργία ενός HMM για κάθε υπογραφή, έτσι ώστε να διαφοροποιηθεί η υπογραφή του από κάθε άλλη υπογραφή. Η τοπολογία των HMM που θα χρησιμοποιηθεί είναι αυτή που ταιριάζει αρκετά στο λατινικό τρόπο γραφής (από αριστερά προς τα δεξιά) και είναι τα αριστερά προς τα δεξιά HMM χωρίς να επιτρέπεται η παράληψη κάποιας κατάστασης. Ο αριθμός των καταστάσεων θα διαφέρει από υπογραφή σε υπογραφή, εφόσον δεν έχουν όλες οι υπογραφές τις ίδιες διαστάσεις. Το καλύτερο μοντέλο επιλέγεται με βάση των πιθανοτικό αλγόριθμο εκπαίδευσης του μοντέλου, όπως αυτός περιγράφηκε στο προηγούμενο κεφάλαιο.

Μετά τον τερματισμό της εκπαιδευτικής διαδικασίας του μοντέλου, αποφασίζεται το κατώφλι αποδοχής και το κατώφλι απόρριψης, με βάση τον χαμηλότερο μέσο όρο False Negative λαθών και False Positive λαθών. Για να ληφθεί υπόψιν και η διαφορά στο μήκος των υπογραφών, κάθε λογάριθμος πιθανότητας που υπολογίζεται ώστε να αποφασισθεί η αποδοχή ή η απόρριψη μιας υπογραφής κανονικοποιείται με βάση το μέγεθος της εκάστοτε υπογραφής.

3.1.1.3 Διαδικασία αυθεντικοποίησης υπογραφών

Για να ελεγχθεί μία υπογραφή θα πρέπει πρώτα να μετατραπεί σε μία ακολουθία παρατηρήσεων σύμφωνα με την διαδικασία εξαγωγής χαρακτηριστικών και στη συνέχεια κάνοντας χρήση του αλγόριθμου Forward-Backward υπολογίζεται η πιθανότητα της συγκεκριμένης ακολουθίας παρατηρήσεων, όπως αυτή προέκυψε από την υπό εξέταση υπογραφή. Η πιθανότητα που υπολογίζεται συγκρίνεται με τα κατώφλια αποδοχής και απόρριψης και ανάλογα βγαίνει το συμπέρασμα για την υπό εξέταση υπογραφή.

3.1.1.4 Πειράματα

Για τον πειραματικό έλεγχο του συστήματος χρησιμοποιήθηκε μία βάση δεδομένων με υπογραφές 100 ανθρώπων, οι οποίοι χωρίστηκαν σε 2 ομάδες των 40 και 60 ατόμων. Η πρώτη ομάδα χρησιμοποιήθηκε για τη δημιουργία του codebook , εκτός από τον έλεγχο και η άλλη ομάδα χρησιμοποιήθηκε μόνο για τον έλεγχο.

Οι 20 υπογραφές από κάθε άνθρωπο χρησιμοποιήθηκαν στη διαδικασία της εκπαίδευσης και οι υπόλοιπες χρησιμοποιήθηκαν στην διαδικασία της αυθεντικοποίησης. Η απόδοση του συστήματος αξιολογείται με βάση τους μέσους όρους των FALSE NEGATIVE, FALSE POSITIVE αποτελεσμάτων.

Τα αποτελέσματα για την πρώτη ομάδα, με βάση την οποία δημιουργήθηκε το codebook είναι σαφώς καλύτερα από την δεύτερη ομάδα η οποία δεν συμμετείχε στη δημιουργία του codebook και επομένως οι υπογραφές των ανθρώπων αυτής της ομάδας δεν είχαν τόσο καλή αναπαράσταση από την κβαντοποίηση των διανυσμάτων που τις αποτελούσαν.

3.1.2 Σύστημα αναγνώρισης υπογραφών με συνεχή HMM

Το συγκεκριμένο σύστημα [2] κάνει χρήση συνεχών HMM για την αναγνώριση υπογραφών, χρησιμοποιώντας χαρακτηριστικά των υπογραφών που εξαρτώνται από τον χρόνο, όπως αυτά συγκεντρώνονται από οθόνες αφής ή ταμπλέτες.

3.1.2.1 Εξαγωγή χαρακτηριστικών

Κάθε υπογραφή αναπαρίσταται σαν μια ακολουθία διανυσμάτων. Κάθε διάνυσμα αποτελείται από μία τιμή για καθεμία από τις εξής χρονοσειρές καθώς επίσης και τη χρονική τους παράγωγο:

- Οριζόντια θέση
- Κάθετη θέση
- Αζιμούθιο
- Εφαπτομένη γωνίας μονοπατιού
- Πλάτος ταχύτητας (Πλάτος διακριτής παραγώγου θέσης)
- Ακτίνα καμπυλότητας
- Πλάτος επιτάχυνσης (Πλάτος διακριτής παραγώγου ταχύτητας)

Οι διακριτές παράγωγοι υπολογίζονται κάνοντας χρήση της μεθόδου second order regression [3].

Το σήμα στη συνέχεια κανονικοποιείται ώστε να αποκτήσει μηδενική μέση τιμή και μοναδιαία τυπική απόκλιση. Το τελικό αποτέλεσμα είναι κάθε υπογραφή να αναπαριστάται από ένα πίνακα που περιέχει 14 κανονικοποιημένες διακριτές χρονοσειρές.

3.1.2.2 Μοντελοποίηση υπογραφών

Για τη δημιουργία ενός HMM ανά είδος υπογραφής, είναι απαραίτητος ο προσδιορισμός των στοιχείων λ=(A,B,π), ο οποίος θα γίνει με βάση το σύνολο των υπογραφών που θα χρησιμοποιηθούν για την εκπαίδευση. Στο συγκεκριμένο σύστημα χρησιμοποιούνται συνεχή HMM, δηλαδή η κατανομή πιθανότητας των συμβόλων παρατήρησης μοντελοποιείται σαν μίξη Κανονικών πολυμεταβλητών κατανομών. Κάθε υπογραφή χωρίζεται σε κομμάτια και κάθε κομμάτι χωρίζεται σε M μέρη με χρήση του αλγόριθμου K-Means [4] , με βάση τα οποία υπολογίζονται οι αρχικές παράμετροι του πίνακα B. Με την επαναληπτική διαδικασία του Baum-Welch υπολογίζεται στη συνέχεια μια καλύτερη εκτίμηση του μοντέλου, έως ότου υπάρχει σύγκλιση.

3.1.2.3 Πειράματα

Για τον έλεγχο του συστήματος χρησιμοποιήθηκε η βάση δεδομένων MCYT οι οποία περιέχει αυθεντικές και επιδέξιες πλαστογραφίες και δημιουργήθηκε ως εξής: Κάθε χρήστης δημιούργησε ένα σύνολο από 25 αυθεντικές υπογραφές και στη συνέχεια 5 επιδέξιες πλαστογραφίες για κάθε ένα από τους προηγούμενους χρήστες. Επομένως για κάθε χρήστη έχουμε 25 αυθεντικές και 25 πλαστογραφημένες υπογραφές.

Διαφορετικό κατώφλι απόφασης επιλέγεται για κάθε χρήστη, εφόσον προκύπτουν καλύτερες επιδόσεις από την επιλογή ενός ενιαίου κατωφλίου για όλους τους χρήστες.

Τα πειράματα που διεξάγονται με την βάση δεδομένων που δημιουργήθηκε καταδεικνύουν σαν ιδανική διαμόρφωση των HMM τις 2 κρυφές καταστάσεις και το συνδυασμό 32 Κανονικών κατανομών ανά κατάσταση για την πυκνότητα κατανομής πιθανότητας των συμβόλων εξόδου. Επιπλέον, από τα πειράματα προκύπτει ότι 5 υπογραφές είναι αρκετές έτσι ώστε να εκπαιδευτεί ικανοποιητικά το μοντέλο για κάθε διαφορετικό τύπο υπογραφής.

Κεφάλαιο 4

ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΗ

4.1 Ανάλυση και Περιγραφή Υποσυστημάτων

Στο παρόν κεφάλαιο δίνεται μια περιγραφή όλων των υποσυστημάτων τα οποία συνθέτουν την εφαρμογή που δημιουργήθηκε. Το σύστημα που δημιουργήθηκε στα πλαίσια αυτής της διπλωματικής είναι μια εφαρμογή για smart-phones με λογισμικό Android η οποία έχει τις εξής λειτουργίες:

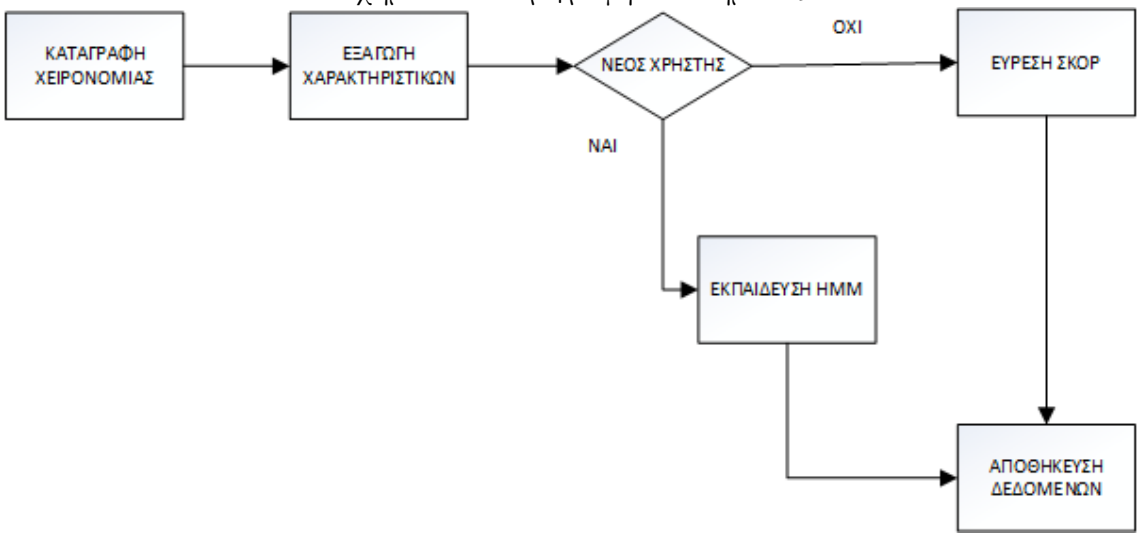
- Συλλογή και αποθήκευση χειρονομιών πάνω στην οθόνη αφής
- Υπολογισμός γεωμετρικών χαρακτηριστικών για κάθε χειρονομία, τα οποία θα χρησιμοποιηθούν για την εκπαίδευση των μοντέλων.
- Δημιουργία νέου χρήστη σε βάση δεδομένων και εκπαίδευση ενός HMM με βάση 10 ίδιες χειρονομίες που εισάγει στο σύστημα.
- Εύρεση ομοιότητας νέων χειρονομιών του κάθε χρήστη με το ήδη υπάρχον HMM, που έχει δημιουργηθεί μοναδικά για αυτόν.

Ακολουθεί η συνοπτική περιγραφή των υποσυστημάτων που υλοποιούν τις παραπάνω λειτουργίες. Οι εφαρμογές που υλοποιούν τις παραπάνω διαδικασίες περιγράφονται αναλυτικά στο επόμενο κεφάλαιο και φαίνονται στην εικόνα 4.1.

4.1.1 Υποσύστημα γραφικής διαπροσωπείας χρήστη

Το υποσύστημα αυτό είναι υπεύθυνο για τη συλλογή των χειρονομιών καθώς και την επεξήγηση των διαφόρων σταδίων της εφαρμογής στο χρήστη με διάφορες υποδείξεις. Αρχικά ο χρήστης θα πρέπει να εγγραφεί στο σύστημα διαλέγοντας ένα 'όνομα χρήστη' το οποίο θα είναι μοναδικό για τον ίδιο και θα χρησιμοποιηθεί από αυτόν αργότερα όταν θα πρέπει να εισάγει τις χειρονομίες ελέγχου. Για την εγγραφή το υποσύστημα παρουσιάζει μια φόρμα εισαγωγής στοιχείων, όπως αυτή φαίνεται στην εικόνα 4.2 Στη συνέχεια ο χρήστης οδηγείται στην οθόνη εισαγωγής της χειρονομίας του, η οποία θα χρησιμοποιηθεί για την εκπαίδευση του HMM, είτε για την εισαγωγή χειρονομιών για τον υπολογισμό του σκορ τους με βάση το ήδη υπάρχον μοντέλο του χρήστη. Οι δύο οθόνες είναι εμφανισιακά ίδιες και για αυτό παρατίθεται το στιγμιότυπο οθόνης μίας από τις δύο στην εικόνα 4.3

Σχήμα 4.1: Περιγραφή Συστήματος



4.1.2 Υποσύστημα εξαγωγής χαρακτηριστικών από τις χειρονομίες

Κάθε χειρονομία πριν χρησιμοποιηθεί για την εκπαίδευση του HMM ή την εύρεση του σκορ της, θα πρέπει να μετασχηματιστεί σε ένα πίνακα διανυσμάτων αποτελούμενα από τα χαρακτηριστικά που έχουμε διαλέξει για την μοντελοποίηση των χειρονομιών. Επομένως μετά από κάθε επιτυχή καταγραφή χειρονομίας ενός χρήστη, οι συντεταγμένες όλων των σημείων που καταγράφηκαν κανονικοποιούνται με βάση το μέγιστο πλάτος και ύψος της οθόνης και στη συνέχεια υπολογίζονται τα εξής χαρακτηριστικά για κάθε σημείο που καταγράφεται:

- γωνίας της κλίσης από το προηγούμενο σημείο
- γωνία από το κέντρο μάζας της χειρονομίας
- απόσταση από το κέντρο μάζας της χειρονομίας

Επιπλέον, για κάθε σημείο που καταγράφεται στην οθόνη έχουμε εξ αρχής την ταχύτητα με την οποία κινείται το δάχτυλο στον κατακόρυφο και στον οριζόντιο άξονα, την πίεση που ασκεί το δάχτυλο και τον αύξοντα αριθμό του δαχτύλου που δημιουργεί την χειρονομία.

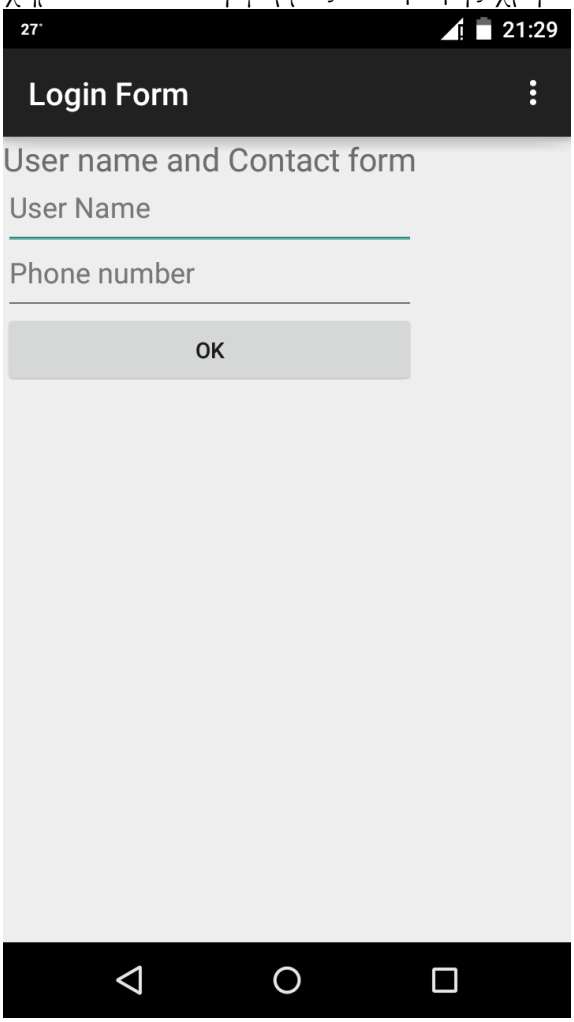
Εφόσον το σύστημα υποστηρίζει χειρονομίες όσων δαχτύλων θέλει ο χρήστης, κάθε σημείο έχει επίσης ένα αύξοντα αριθμό για το δάχτυλο το οποίο δημιούργησε το σημείο αυτό, ξεκινώντας από το 0 για το πρώτο δάχτυλο που ακούμπησε την οθόνη και αυξάνοντας κατά 1 κάθε φορά που ένα νέο δάχτυλο ακουμπά την οθόνη.

Τελικά, κάθε σημείο που καταγράφεται από την οθόνη αφής μετατρέπεται σε ένα διάνυσμα $\vec{o}$ 7 στοιχείων, ως εξής

$$\vec{o} = [i, v_x, v_y, p, \theta_1, \theta_2, loc]$$

όπου i αντιπροσωπεύει τον αύξων αριθμό του δαχτύλου που δημιούργησε το σημείο, v_x, v_y αντιπροσωπεύουν τις ταχύτητες στον οριζόντιο άξονα x και στον κατακόρυφο

Σχήμα 4.2: Οθόνη φόρμας εισαγωγής χρήστη



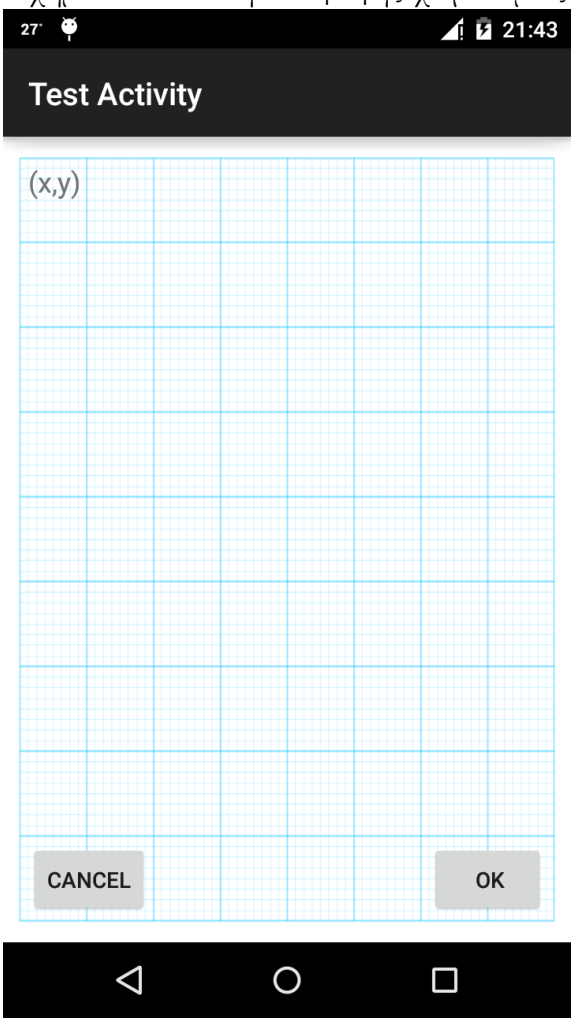
άξονα y αντίστοιχα, p αντιπροσωπεύει την πίεση που ασκείται από το δάχτυλο στο συγκεκριμένο σημείο, θ_1 αντιπροσωπεύει τη γωνία του σημείου από το κέντρο μάζας της χειρονομίας, θ_2 αντιπροσωπεύει τη γωνία από το προηγούμενο σημείο και η loc αντιπροσωπεύει την απόσταση του σημείου από το κέντρο μάζας.

4.1.3 Υποσύστημα εκπαίδευσης του HMM

Για να δημιουργηθεί ένα HMM που θα χαρακτηρίζει μοναδικά κάθε χρήστη, είναι απαραίτητη η εκπαίδευση του με 10 όμοιες χειρονομίες του ίδιου χρήστη. Το υποσύστημα εκπαίδευσης δέχεται τις χειρονομίες οι οποίες είναι πλέον μετασχηματισμένες σε πίνακα με διανύσματα παρατήρησης και είναι υπεύθυνο για την σωστή εκπαίδευση του μοντέλου, καθώς επίσης και για την σωστή αρχικοποίηση του υπό εκπαίδευση μοντέλου.

Το υποσύστημα αυτό βασίζεται στις υλοποιημένες συναρτήσεις της βιβλιοθήκης Jahmm [5] με κάποιες τροποποιήσεις λόγων ορισμένων σφαλμάτων που ανακαλύφθηκαν κατά την ανάπτυξη της εφαρμογής.

Σχήμα 4.3: Οθόνη εισαγωγής χειρονομίας



4.1.4 Υποσύστημα αποθήκευσης δεδομένων σε σχεσιακή βάση δεδομένων

Εφόσον το ζητούμενο είναι ο κάθε χρήστης να επιστρέφει στην εφαρμογή για να εισάγει νέες χειρονομίες, θα πρέπει να υπάρχει ένα σύστημα το οποίο θα διαχειρίζεται την αλληλεπίδραση της εφαρμογής με μία βάση δεδομένων η οποία θα περιέχει στοιχεία του κάθε χρήστη.

Το υποσύστημα αυτό είναι υπεύθυνο για την δημιουργία νέου χρήστη στο σύστημα, την αποθήκευση των στοιχείων του και του μοντέλου που θα τον χαρακτηρίζει καθώς και για την αποθήκευση όλων των χειρονομιών του κάθε χρήστη.

Επιπλέον, το σύστημα είναι υπεύθυνο για να φέρνει από τη βάση δεδομένων το αποθηκευμένο μοντέλο κάθε χρήστη όταν θα πρέπει να γίνει η εύρεση του σκορ για κάθε νέα χειρονομία.

4.1.5 Υποσύστημα εύρεσης πιθανότητας της χειρονομίας με δεδομένο το HMM του χρήστη

Για την επαλήθευση της λειτουργίας του συστήματος θα χρειαστεί ο υπολογισμός ενός σκορ για νέες χειρονομίες του χρήστη με δεδομένο το HMM του. Επομένως είναι απαραίτητο ένα υποσύστημα το οποίο θα παίρνει τις καταγεγραμμένες χειρονομίες του χρήστη, οι οποίες προηγουμένως έχουν μετασχηματιστεί σε πίνακα διανυσμάτων παρατήρησης μέσω του υποσυστήματος εξαγωγής χαρακτηριστικών, και θα υπολογίζει την πιθανότητα να έχουν προέλθει από το μοντέλο του χρήστη.

Το υποσύστημα θα χρησιμοποιεί τις συναρτήσεις της βιβλιοθήκης Jahmm έτσι ώστε να υπολογιστεί το σκορ κάθε χειρονομίας.

Κεφάλαιο 5

ΥΛΟΠΟΙΗΣΗ

Σε αυτό το κεφάλαιο γίνεται αναλυτική περιγραφή της εφαρμογής. Αρχικά, γίνεται μία αναφορά στην πλατφόρμα, τα προγραμματιστικά εργαλεία και τις βιβλιοθήκες που χρησιμοποιήθηκαν. Στη συνέχεια περιγράφονται αναλυτικά οι κλάσεις της εφαρμογής.

5.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Στην ενότητα αυτή περιγράφονται τα χαρακτηριστικά της συγκεκριμένης υλοποίησης, όπως η πλατφόρμα ανάπτυξης και εκτέλεσης, τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν καθώς και οι απαιτήσεις της εφαρμογής.

5.1.1 Εργαλεία ανάπτυξης

Η εφαρμογή αναγνώρισης των χειρονομιών σε οθόνη αφής αναπτύχθηκε εξ΄ ολοκλήρου σε java version "1.7.0-60". Η κύρια πλατφόρμα ανάπτυξης, η οποία στηρίχθηκε στα Android 5.0 SDK Platform και Jahmm library ήταν το Android Studio version 1.3.1. Διορθώσεις στη βιβλιοθήκη Jahmm library έγιναν με το Eclipse IDE. Για την αποθήκευση των δεδομένων χρησιμοποιήθηκε η SQLite database engine , ενώ για τη στατιστική επεξεργασία των αποτελεσμάτων και την παραγωγή διαγραμμάτων χρησιμοποιήθηκε το πρόγραμμα MATLAB.

5.1.2 Πλατφόρμα ανάπτυξης

Στην πλατφόρμα Android Studio version 1.3.1. αναπτύχθηκε το γραφικό περιβάλλον, όπως αυτό περιγράφεται στο προηγούμενο κεφάλαιο, καθώς επίσης και το υποσύστημα αποθήκευσης και ανάκτησης δεδομένων. Η συγκεκριμένη πλατφόρμα είναι το επίσημο περιβάλλον ανάπτυξης Android εφαρμογών και προτείνεται από την Google. Το συγκεκριμένο εργαλείο παρέχει τις εξής δυνατότητες:

- Γραφικός σχεδιασμός φορμών και οθονών αλληλεπίδρασης με το χρήστη.
- Ενοποίηση με το version control system Github για την ασφαλή αποθήκευση και επιθεώρηση των διαφορετικών εκδόσεων του κώδικα καθ΄ όλη τη διάρκεια

ανάπτυξης της εφαρμογής.

- Εύχρηστο σύστημα για αποσφαλμάτωση των προγραμμάτων που κάνει πιο γρήγορη τη διαδικασία ανάπτυξης.
- Εύκολη πρόσβαση σε βιβλιοθήκες Android 5.0 SDK Platform οι οποίες είναι απαραίτητες για την ανάπτυξη της εφαρμογής.

Διόρθωση σφαλμάτων που υπήρχαν στη Jahmm library έγιναν με το εργαλείο Eclipse IDE, το οποίο είναι από τα πιο διαδεδομένα περιβάλλοντα ανάπτυξης εφαρμογών Java και παρέχει ανάλογες ευκολίες κατά την ανάπτυξη εφαρμογών, όπως και το Android Studio

5.1.3 Εγκατάσταση και εκτέλεση

Η εφαρμογή μπορεί να εκτελεστεί σε κινητά με λογισμικό Android 3.2 Honeycomb έωςAndroid 5.0 Lollipop και για την εγκατάστασή της πρέπει να ακολουθήσουμε τα εξής βήματα, έτσι ώστε να επιτρέψουμε την εγκατάσταση εφαρμογών που δεν προέρχονται από το επίσημο κατάστημα της Google:

- Μεταφερόμαστε στις ρυθμίσεις της συσκευής και επιλέγουμε ¨Ρυθμίσεις Ασφαλείας (Security Settings)"
- Στη συνέχεια επιλέγουμε "Αγνωστες Πηγές(Unknown Sources)" και επιλέγουμε το πεδίο αποδεχόμενοι την προειδοποίηση που θα εμφανιστεί.

Στη συνέχεια ανοίγουμε το αρχείο .apk της εφαρμογής, το οποίο πρέπει να έχει μεταφερθεί στη συσκευή και η εφαρμογή εγκαθίσταται αυτόματα. Για να εκτελέσουμε την εφαρμογή απλά πατάμε πάνω στο εικονίδιο της εφαρμογής που θα εμφανιστεί αμέσως μετά την εγκατάσταση.

5.2 Βιβλιοθήκες

5.2.1 Jahmm library

Η βιβλιοθήκη [5] παρέχει μία υλοποίηση των HMM σε Java και οι λόγοι για τους οποίους επιλέχθηκε είναι το γεγονός ότι παρέχει υλοποιήσεις αλγορίθμων που θα χρειαστούμε για την εκπαίδευση και τον έλεγχο των HMM και κυρίως γιατί είναι γραμμένη σε Java η οποία είναι η γλώσσα προγραμματισμού για την ανάπτυξη εφαρμογών σε κινητά Android. Επομένως η επιλογή μας η τελική εφαρμογή να παρουσιαστεί σε ένα κινητό Android ήταν δεσμευτική ως προς το ποια βιβλιοθήκη θα χρησιμοποιούσαμε και ενδεχομένως να υπάρχουν καλύτερες υλοποιήσεις σε διαφορετικές γλώσσες προγραμματισμού.

Η βιβλιοθήκη έχει αναπτυχθεί από τον J.M Francois και αποτελείται από τα εξής πακέτα:

- Distributions: το οποίο παρέχει υλοποιήσεις διάφορων κατανομών πιθανότητας που μπορούν να χρησιμοποιηθούν με συνεχή HMM. Πιο συγκεκριμένα παρέχει υλοποιήσεις για την Εκθετική κατανομή, την Γκαουσιανή κατανομή, την κατανομή

Poisson την μίξη Γκαουσιανών κατανομών και την πολυμεταβλητή Γκαουσιανή κατανομή. Παρέχει επίσης και διαπροσωπείες για την υλοποίηση Διακριτών κατανομών και τυχαίων πολυμετάβλητων κατανομών.

- Jahmm: το πακέτο αυτό περιέχει τις κύριες κλάσεις που θα μας απασχολήσουν και οι οποίες θα αναλυθούν παρακάτω. Πιο συγκεκριμένα περιέχει κλάσεις που υλοποιούν τα HMM, καθώς και τα διαφορετικά είδη παρατηρήσεων.

- Draw: το πακέτο περιέχει την κλάση GenericHmmDrawerDot για την απεικόνιση των HMM σε .dot αρχεία. Το πακέτο αυτό δεν θα μας απασχολήσει.

- Io: το πακέτο αυτό περιέχει κλάσεις που χρησιμεύουν για το input-output όλων των αντικειμένων της βιβλιοθήκης που θα μας απασχολήσουν. Οι κλάσεις του συγκεκριμένου πακέτου θα χρησιμοποιηθούν για την εγγραφή των HMM στη βάση δεδομένων και την ανάκτηση τους από αυτή.

- Learn: το πακέτο αυτό περιέχει τις κλάσεις που θα χρησιμοποιηθούν για την εκπαίδευση των μοντέλων, δηλαδή παρέχει υλοποιήσεις για τους Baum-Welch, Kmeans αλγόριθμους.

- Toolbox: το πακέτο περιέχει διάφορα εργαλεία που σχετίζονται με τα HMM, όπως την κλάση KullbackLeiblerDistanceCalculator που υπολογίζει την απόσταση μεταξύ δύο HMM, και την κλάση MarkovGenerator η οποία παρέχει ακολουθίες που θα μπορούσε να παράγει ένα μοντέλο. Το συγκεκριμένο πακέτο δεν θα μας απασχολήσει.

Ακολουθεί η παρουσίαση των κλάσεων που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.

5.2.1.1 class MultiGaussianDistribution

Η κλάση υλοποιεί μια πολυμεταβλητή Γκαουσιανή κατανομή. Θα χρησιμοποιηθεί για την μοντελοποίηση των συναρτήσεων κατανομής των συμβόλων παρατήρησης. Εφόσον τα διανύσματα παρατήρησης έχουν μέγεθος 7, ο κατασκευαστής της κλάσης θα είναι ο εξής:

```
MultiGaussianDistribution(int dimension)
```

ο οποίος δημιουργεί μία νέα πολυμεταβλητή Γκαουσιανή κατανομή με μηδενική μέση τιμή και μοναδιαία διακύμανση. Οι μέθοδοι της κλάσης είναι οι εξής:

- `double[][] covariance();`

  πίνακας διασποράς της κατανομής

- `double covarianceDet();`

  ορίζουσα του πίνακα διασποράς της κατανομής

- `int dimension() ;`

  διάσταση της κατανομής

- **double**[] generate() ;

  ψευδοτυχαίο διάνυσμα με βάση την κατανομή

- **double**[] mean() ;

  τον πίνακα μέσων τιμών της κατανομής

- **double** probability(**double**[] v) ;

  την πυκνότητα πιθανότητας ενός δοσμένου διανύσματος

5.2.1.2 class Hmm<O extends Observation>

Είναι η κύρια HMM κλάση και υλοποιεί ένα Hidden Markov Model το οποίο αποτελείται από

- καταστάσεις: κάθε κατάσταση έχει δοσμένη μία πιθανότητα αρχικής κατάστασης (π) και είναι συσχετισμένη με συνάρτηση πυκνότητας πιθανότητας συμβόλων παρατήρησης (*opdf*). Κάθε κατάσταση σχετίζεται με ένα δείκτη΄ η πρώτη κατάσταση αριθμείται 0 και η τελευταία $n-1$ (όπου n είναι το πλήθος των καταστάσεων του HMM). Αυτός ο δείκτης περνιέται σαν όρισμα σε διάφορες συναρτήσεις για οποιαδήποτε αναφορά στη συγκεκριμένη κατάσταση.

- πιθανότητες μετάβασης: η πιθανότητα μεταφοράς από την κατάσταση i στην κατάσταση j (a_{ij}).

ο κατασκευαστής της κλάσης θα είναι ο εξής:

```
Hmm<ObservationVector>(nbStates, new OpdfMultiGaussianFactory
        (int dimension));
```

ο οποίος δημιουργεί ένα νέο HMM με *nbStates* αριθμό καταστάσεων και παρατηρήσεις σε μορφή διανύσματος. Επιπλέον οι κατανομή πυκνότητας πιθανότητας των συμβόλων εξόδου είναι πολυμεταβλητή κατανομή με 7 διαστάσεις. Οι μέθοδοι της κλάσης είναι οι εξής:

- **double** getAij(**int** i, **int** j) ;

  επιστρέφει την πιθανότητα μετάβασης από την κατάσταση i στην κατάσταση j

- Opdf<O> getOpdf(**int** stateNb) ;

  επιστρέφει την συνάρτηση πυκνότητας πιθανότητας της δεδομένης κατάστασης

- **double** getPi(**int** stateNb) ;

  επιστρέφει την πιθανότητα η δεδομένη κατάσταση να είναι αρχική κατάσταση

- **double** lnProbability(java.util.List<? **extends** O> oseq);

  επιστρέφει τον λογάριθμο της πιθανότητας να έχει προκύψει από το HMM η δοσμένη λίστα παρατηρήσεων

- `int[] mostLikelyStateSequence(java.util.List<? extends O> oseq)`

  επιστρέφει ένα πίνακα με την πιο πιθανή ακολουθία καταστάσεων που ταιριάζει σε μια ακολουθία παρατηρήσεων, δεδομένου του HMM

- `int nbStates();`

  τον αριθμό καταστάσεων ενός HMM

- `double probability(java.util.List<? extends O> oseq)`

  την πιθανότητα μιας ακολουθίας παρατηρήσεων δεδομένου του HMM

- `java.lang.String toString()`

  επιστρέφει μια συμβολοσειρά που περιγράφει το HMM

Σημαντικά αντικείμενα που σχετίζονται με την κλάση HMM είναι τα αντικείμενα της κλάσης Observation η οποία μοντελοποιεί τις παρατηρήσεις και μπορεί να χρησιμοποιηθεί είτε σε λίστες είτε σε σύνολα λιστών. Η κλάση θα αναλυθεί στη συνέχεια.

5.2.1.3 class ObservationVector

Η κλάση επεκτείνει την κλάση της απλής παρατήρησης O και υλοποιεί μία παρατήρηση που περιγράφεται από ένα διάνυσμα πραγματικών αριθμών. Εφόσον κάθε παρατήρηση της χειρονομίας είναι ένα διάνυσμα 7 πραγματικών αριθμών, η συγκεκριμένη μορφή παρατήρησης είναι η ιδανική για την εφαρμογή.

ο κατασκευαστής της κλάσης θα είναι ο εξής:

`ObservationVector(double[] value);`

ο οποίος δημιουργεί ένα αντικείμενο που μοντελοποιεί μια παρατήρηση που περιγράφεται από το διάνυσμα πραγματικών αριθμών *value*. Οι μέθοδοι της κλάσης είναι οι εξής:

- `double[] values();`

  επιστρέφει τις τιμές που συνθέτουν το διάνυσμα παρατήρησης

- `java.lang.String toString()`

  επιστρέφει μια συμβολοσειρά που περιγράφει το διάνυσμα παρατήρησης.

5.2.1.4 class ForwardBackwardScaledCalculator

Η κλάση αυτή μπορεί να χρησιμοποιηθεί για να υπολογιστεί η πιθανότητα μια δοσμένης ακολουθίας για ένα συγκεκριμένο HMM. Η κλάση αυτή υλοποιεί τη μέθοδο κλιμάκωσης, όπως αυτή περιγράφεται στην εργασία του Rabiner, Fundamentals of speech recognition [6]. Ο κατασκευαστής της κλάσης είναι ο εξής:

`ForwardBackwardScaledCalculator(java.util.List<? extends O> oseq, Hmm<O> hmm)`

ο οποίος υπολογίζει την πιθανότητα να παρατηρηθεί η ακολουθία παρατήρησης δεδομένου του HMM. Οι μέθοδοι της κλάσης είναι οι εξής:

- **protected**
  <O **extends** Observation>
  void
  computeAlpha(Hmm<? **super** O> hmm, java.util.List<O> oseq) ;

  υπολογίζεται ο πίνακας *alpha*

- **protected**
  <O **extends** Observation>
  void
  computeBeta(Hmm<? **super** O> hmm, java.util.List<O> oseq)

  υπολογίζεται ο πίνακας *beta*.

- **double** lnProbability()

  επιστρέφει το νεπέριο λογάριθμο της πιθανότητας της ακολουθίας που παρήγαγε το αντικείμενο.

5.2.1.5 class KMeansCalculator

Η κλάση αυτή μπορεί να χρησιμοποιηθεί για να χωρίσει ένα σύνολο στοιχείων σε συστάδες χρησιμοποιώντας τον αλγόριθμο k-Means όπως αυτός περιγράφεται στο βιβλίο Clustering and the Continuous k-Means Algorithm (Vance Faber, Los Alamos Science number 22) [7] Ο κατασκευαστής της κλάσης είναι ο εξής:

```
KMeansCalculator(int k, java.util.List<? extends K> elements) ;
```

ο οποίος διαχωρίζει ένα σύνολο στοιχείων σε δοσμένο αριθμό συστάδων.

Η κλάση αυτή είναι αρκετά σημαντική στην υλοποίηση της εφαρμογής, καθώς χρησιμοποιείται κατά τον υπολογισμό του αρχικού HMM, έτσι ώστε να βεβαιωθούμε ότι το μοντέλο θα συγκλίνει κατά τη διαδικασία της εκπαίδευσης.

5.2.1.6 class HmmBinaryReader

Η κλάση αυτή μπορεί να διαβάσει ένα HMM από μία ροή bytes (byte stream) και θα χρησιμοποιηθεί για την ανάγνωση του HMM από την βάση δεδομένων. Ο κατασκευαστής της κλάσης είναι ο εξής:

```
HmmBinaryReader() ;
```

η μέθοδος της κλάσης είναι η εξής:

- **static** Hmm<?> read(java.io.InputStream stream)

  η οποία διαβάζει ένα HMM από ένα byte stream

5.2.1.7 class HmmBinaryWriter

Η κλάση αυτή μπορεί να γράψει ένα HMM σε μία ροή bytes (byte stream) και θα χρησιμοποιηθεί για την εγγραφή του HMM στην βάση δεδομένων. Ο κατασκευαστής της κλάσης είναι ο εξής:

```
HmmBinaryWriter() ;
```

η μέθοδος της κλάσης είναι η εξής:

- **static void** write(java.io.OutputStream stream, Hmm<?> hmm)

  η οποία γράφει ένα HMM σε ένα byte stream

5.2.1.8 class HmmReader

Η κλάση αυτή διαβάζει ένα HMM το οποίο είναι γραμμένο σε κανονική μορφή και όχι σε μορφή bytes. Ο κατασκευαστής της κλάσης είναι ο εξής:

```
HmmReader() ;
```

η μέθοδος της κλάσης είναι η εξής:

- **static** <O **extends** Observation> Hmm<O> read(java.io.Reader reader, OpdfReader<? **extends** Opdf<O>> opdfReader)

  η οποία διαβάζει ένα HMM γραμμένο με λατινικούς χαρακτήρες

5.2.1.9 class HmmWriter

Η κλάση αυτή γράφει ένα HMM σε λατινικούς χαρακτήρες και σε μορφή που να μπορεί να διαβαστεί από την κλάση HmmReader. Ο κατασκευαστής της κλάσης είναι ο εξής:

```
HmmWriter() ;
```

η μέθοδος της κλάσης είναι:

- **static** <O **extends** Observation> **void** write(java.io.Writer writer, OpdfWriter<? **extends** Opdf<O>> opdfWriter, Hmm<O> hmm)

  η οποία γράφει ένα HMM με λατινικούς χαρακτήρες και στη μορφή που πρέπει

5.2.1.10 class BaumWelchScaledLearner

Η κλάση αυτή είναι μια υλοποίηση του αλγόριθμου εκπαίδευσης Baum-Welch και χρησιμοποιεί ένα μηχανισμό κλιμάκωσης, έτσι ώστε να αποφεύγονται φαινόμενα υποχείλισης. Ο κατασκευαστής της κλάσης είναι ο εξής:

```
BaumWelchScaledLearner()
```

και οι μέθοδοι της κλάσης είναι οι εξής:

- **protected** <O **extends** Observation> **double**[][][] estimateXi(java.util.List<? **extends** O> sequence, ForwardBackwardCalculator fbc, Hmm<O> hmm)

  protected <O **extends** Observation> ForwardBackwardCalculator generateForwardBackwardCalculator(java.util.List<? **extends** O> sequence, Hmm<O> hmm)

5.2.1.11 class KMeansLearner

Η κλάση αυτή αρχικοποιεί την υλοποίηση ενός αλγορίθμου K-Means, ο οποίος βρίσκει ένα HMM το οποίο μοντελοποιεί ένα σύνολο παρατηρήσεων. Ο κατασκευαστής της κλάσης είναι ο εξής:

KMeansLearner(**int** nbStates, OpdfFactory<? **extends** Opdf<O>> opdfFactory, java.util.List<? **extends** java.util.List<? **extends** O>> sequences)

και οι μέθοδοι της κλάσης είναι οι εξής:

- **boolean** isTerminated()

  η οποία επιστρέφει true εάν ο αλγόριθμος φτάσει την επιθυμητή ακρίβεια, αλλιώς επιστρέφει false

- Hmm<O> iterate()

  η οποία εκτελεί μία επανάληψη του αλγόριθμου K-Means

- Hmm<O> learn()

  η οποία εκτελεί επαναλήψεις του αλγόριθμου K-Means μέχρι να επιτευχθεί ένα επίπεδο ακρίβειας.

5.2.2 Android Libraries

5.2.2.1 class MotionEvent

Τα αντικείμενα αυτής της κλάσης χρησιμοποιούνται για να αναφέρουν συμβάντα κίνησης. Τα συμβάντα αυτά μπορούν να περιέχουν απόλυτες ή σχετικές κινήσεις καθώς και άλλου τύπου δεδομένα, ανάλογα με τη συσκευή.

Τα συμβάντα κίνησης περιγράφουν κινήσεις με βάση ένα κωδικό ενέργειας και ένα σύνολο τιμών αξόνων. Ο κωδικός ενέργειας προσδιορίζει την αλλαγή κατάστασης που συνέβη, όπως ένας δείκτης να ακούμπησε την οθόνη ή να απομακρύνθηκε από αυτή. Οι τιμές των αξόνων περιγράφουν την κίνηση και άλλες ιδιότητες της κίνησης.

Για παράδειγμα, όταν ένας χρήστης ακουμπά για πρώτη φορά την οθόνη, το σύστημα παραδίδει ένα συμβάν κίνησης στην κατάλληλη προβολή με τον κωδικό κίνησης $ACTION - DOWN$ και ένα σύνολο τιμών αξόνων που περιέχουν τις X και Υ συντεταγμένες της επαφής και πληροφορίες για την πίεση, το μέγεθος και τον προσανατολισμό της επιφάνειας επαφής.

Αρκετές συσκευές μπορούν να αναφέρουν πολλαπλά ίχνη κίνησης την ίδια στιγμή. Οι συγκεκριμένες συσκευές εκπέμπουν ένα ίχνος κίνησης για κάθε δάχτυλο. Το κάθε δάχτυλο αναφέρεται σαν pointer. Τα Motion Events περιέχουν πληροφορίες για όλους τους pointers που είναι ακόμα ενεργοί, ακόμα και αν δεν έχουν κουνηθεί από το προηγούμενο συμβάν.

Το πλήθος των pointers αλλάζει μόνο κατά ένα καθώς μεμονωμένοι δείκτες ανεβοκατεβαίνουν στην οθόνη, εκτός κι εάν ακυρωθεί η χειρονομία. Κάθε pointer έχει μία μοναδική ταυτότητα η οποία του δίνεται όταν ακουμπά για πρώτη φορά την οθόνη (με τα events $ACTION - DOWN$, $ACTION - POINTER - DOWN$) . Η ταυτότητα παραμένει μέχρι να σταματήσει να ακουμπά την οθόνη ο δείκτης (με τα events $ACTION - UP$, $ACTION - POINTER - UP$) ή όταν ακυρώνεται η χειρονομία (με το event $ACTION - CANCEL$) .

Οι μέθοδοι της βιβλιοθήκης της οποίες θα χρησιμοποιήσουμε είναι οι εξής:

- **public final int** getActionIndex ()

  επιστρέφει την ταυτότητα του pointer που προκάλεσε το συμβάν κίνησης

- **public final float** getX (**int** pointerIndex)

  επιστρέφει τη X συντεταγμένη για την δοσμένη ταυτότητα pointer. Οι ακέραιοι αριθμοί είναι pixels. Συσκευές οι οποίες έχουν μεγαλύτερη ακρίβεια από ένα pixel μπορεί να επιστρέψουν κλασματικές τιμές.

- **public final float** getY (**int** pointerIndex)

  επιστρέφει τη Υ συντεταγμένη για την δοσμένη ταυτότητα pointer. Οι ακέραιοι αριθμοί είναι pixels. Συσκευές οι οποίες έχουν μεγαλύτερη ακρίβεια από ένα pixel μπορεί να επιστρέψουν κλασματικές τιμές.

- **public final float** getPressure (**int** pointerIndex)

  επιστρέφει την στιγμιαία πίεση για την δεδομένη ταύτοτητα pointer. Η πίεση παίρνει τιμές από 0 (καθόλου πίεση) μέχρι 1 (κανονική πίεση). Ενδεχομένως να παρουσιαστούν τιμές μεγαλύτερες του 1, ανάλογα με την βαθμονόμηση της συσκευής εισόδου.

5.2.2.2 class VelocityTracker

Τα αντικείμενα αυτής της κλάσης βοηθούν στην παρακολούθηση της ταχύτητας των συμβάντων κίνησης. Για να χρησιμοποιηθεί ένα τέτοιο αντικείμενο πρέπει να ανακτήσουμε ένα νέο στιγμιότυπο του και να του προσθέσουμε στη συνέχεια τα συμβάντα κίνησης που λαμβάνουμε. Στη συνέχεια καλούμε τη μέθοδο υπολογισμού της ταχύτητας και λαμβάνουμε τις ταχύτητες στον οριζόντιο και στον κατακόρυφο άξονα για το δεδομένο pointer.

Οι μέθοδοι της κλάσεις που θα χρησιμοποιήσουμε είναι οι εξής:

- **public static** VelocityTracker obtain ()

Ανάκτηση ενός νέου αντικειμένου VelocityTracker για να παρακολουθείται η ταχύτητα

- **public void** recycle ()

  επιστροφή του αντικειμένου VelocityTracker για να μπορέσει να χρησιμοποιηθεί από άλλους.

- **public void** addMovement (MotionEvent event)

  προσθέτει την κίνηση ενός χρήστη στον αντικείμενο της κλάσης, έτσι να παρακολουθείται η ταχύτητα της κίνησης. Η μέθοδος αυτή θα ήταν καλό να καλείται για το αρχικό συμβάν κίνησης $ACTION - DOWN$, το αμέσως επόμενο συμβάν $ACTION - MOVE$ και το τελικό συμβάν $ACTION - UP$.

- **public void** computeCurrentVelocity (**int** units)

  υπολογίζεται η στιγμιαία ταχύτητα βασισμένη στα σημεία που έχουν συγκεντρωθεί. Η μέθοδος θα πρέπει να καλείται μόνο όταν θέλουμε να υπολογίσουμε την ταχύτητα καθώς είναι σχετικά δαπανηρή ενέργεια.

- **public float** getXVelocity (**int** id)

  επιστροφή της τελευταίας υπολογισμένης ταχύτητας του δοσμένου pointer στον άξονα X. Πρέπει πάντα να προηγείται η κλήση της μεθόδου computeCurrentVelocity(int).

- **public float** getYVelocity (**int** id)

  επιστροφή της τελευταίας υπολογισμένης ταχύτητας του δοσμένου pointer στον άξονα Ψ. Πρέπει πάντα να προηγείται η κλήση της μεθόδου computeCurrentVelocity(int).

5.2.2.3 class AsyncTask

Η κλάση AsyncTask καθιστά ικανή τη σωστή και εύκολη χρήση του νήματος γραφικής διαπροσωπείας (UI thread). Η κλάση αυτή επιτρέπει την πραγματοποίηση εργασιών στο παρασκήνιο και την δημοσιοποίηση τους στο νήμα της γραφικής διαπροσωπείας χωρίς να είναι απαραίτητη η διαχείριση νημάτων ή/και χειριστών (handlers).

Η AsyncTask είναι σχεδιασμένη σαν βοηθητική κλάση γύρω από τις κλάσεις Thread, Handler και δεν αποτελεί ένα γενικευμένο πλαίσιο νηματοποίησης. Ιδανικά θα πρέπει να χρησιμοποιείται για μικρές εργασίες (λίγων δευτερολέπτων το πολύ).

Μία ασύγχρονη εργασία (asynchronous task) ορίζεται από ένα υπολογισμό που τρέχει στο νήμα παρασκηνίου και του οποίου το αποτέλεσμα δημοσιοποιείται στο νήμα γραφικής διαπροσωπείας. Ορίζεται από 3 γενικούς τύπους (generic types) που ονομάζονται Params, Progress, Result και από 4 βήματα εκτέλεσης, που ονομάζονται onPreExecute, doInBackground, onProgressUpdate, onPostExecute. Ακολουθεί η περιγραφή των 3 γενικών τύπων και των 4 βημάτων:

Γενικοί τύποι

1. Params: ο τύπος των παραμέτρων που στέλνονται στην εργασία κατά την εκτέλεση

2. Progress: ο τύπος των μονάδων προόδου που δημοσιεύονται κατά τον υπολογισμό στο παρασκήνιο.

3. Result: ο τύπος του αποτελέσματος του υπολογισμού στο παρασκήνιο.

Δεν χρειάζεται να χρησιμοποιηθούν όλοι οι τύποι από μία ασύγχρονη εργασία.

Βήματα εκτέλεσης

1. onPreExecute()

   καλείται από το νήμα γραφικής διαπροσωπείας πριν εκτελεστεί η εργασία. Το βήμα αυτό χρησιμοποιείται συνήθως για να ρυθμιστεί η εργασία, για παράδειγμα για να δείξουμε μια μπάρα προόδου στην διεπαφή του χρήστη.

2. doInBackground(Params...)

   καλείται από το νήμα παρασκηνίου αμέσως μετά η μέθοδος onPreExecute() ολοκληρωθεί. Το βήμα αυτό χρησιμοποιείται για να γίνουν υπολογισμοί παρασκηνίου που θα χρειαστούν αρκετή ώρα. Οι παράμετροι της ασύγχρονης εργασίας εισέρχονται σε αυτό το βήμα. Το αποτέλεσμα του υπολογισμού πρέπει να επιστραφεί από αυτό το βήμα και να περαστεί στο τελευταίο βήμα. Το βήμα αυτό μπορεί επίσης να χρησιμοποιήσει και τη μέθοδο publishProgress(Progress ...) έτσι ώστε να δημοσιοποιηθεί μία ή περισσότερες μονάδες προόδου. Οι τιμές αυτές περνιούνται στο νήμα διαπροσωπείας του χρήστη κατά το βήμα onProgressUpdate(Progress...).

3. onProgressUpdate(Progress...):

   καλείται από το νήμα διαπροσωπείας χρήστη μετά από κλήση της μεθόδου publishProgress(Progress...). Ο χρονισμός της εκτέλεσης είναι απροσδιόριστος. Η μέθοδος χρησιμοποιείται για να απεικονιστεί οποιαδήποτε πληροφορία προόδου στην διαπροσωπεία του χρήστη, όσο το νήμα παρασκηνίου εκτελείται. Για παράδειγμα μπορεί να χρησιμοποιηθεί για να απεικονιστεί μια μπάρα προόδου ή να απεικονιστούν κείμενα σε ένα πεδίο κειμένου.

4. onPostExecute(Result):

   καλείται από το νήμα διαπροσωπείας του χρήστη μετά την ολοκλήρωση του νήματος παρασκηνίου. Το αποτέλεσμα του υπολογισμού περνιέται σαν παράμετρος σε αυτό το βήμα.

5.2.3 SQLite library

Η SQLite είναι μία βιβλιοθήκη η οποία υλοποιεί μία αυτόνομη, χωρίς εξυπηρετητή, με μηδενικές παραμέτρους SQL database engine. Η SQLite είναι μια ολοκληρωμένη SQL database engine η οποία σε αντίθεση με τις περισσότερες SQL βάσεις δεδομένων δεν έχει ξεχωριστή διαδικασία εξυπηρέτησης. Η SQLite γράφει κατευθείαν σε συνηθισμένα αρχεία δίσκου. Μια ολοκληρωμένη SQL βάση με διαφορετικούς πίνακες, triggers, και όψεις περιέχεται σε ένα μοναδικό αρχείο δίσκου. Το file format της βάσης είναι πολλαπλής πλατφόρμας. Οι κλάσεις που θα χρησιμοποιήσουμε για την διαχείριση της βάσης δεδομένων με τους χρήστες και τις χειρονομίες τους περιέχονται στο πακέτο android.database.sqlite περιγράφονται παρακάτω:

5.2.3.1 class SQLiteDatabase

παρέχει μεθόδους για τη διαχείριση μιας SQLite βάσης δεδομένων. Περιέχει μεθόδους για τη δημιουργία, διαγραφή και εκτέλεση SQL εντολών και την εκτέλεση άλλων παρόμοιων εργασιών. οι μέθοδοι που θα χρησιμοποιήσουμε είναι οι εξής:

- **public void** execSQL (String sql)

  εκτελεί μία SQL δήλωση η οποία δεν είναι SELECT ή οποιαδήποτε άλλη εντολή που επιστρέφει δεδομένα.

- **public** Cursor rawQuery (String sql, String[] selectionArgs)

  εκτελεί τη δοσμένη SQL δήλωση και επιστρέφει ένα αντικείμενο Cursor που περιέχει το σύνολο των αποτελεσμάτων.

5.3 Λεπτομέρειες υλοποίησης

5.3.1 class GesturePoint

Αυτή είναι από τις σημαντικότερες κλάσεις της υλοποίησης, καθώς μοντελοποιεί τα σημεία των χειρονομιών. Οι μεταβλήτες της κλάσης αποτελούν τα χαρακτηριστικά που θα χρησιμοποιηθούν για την εκπαίδευση των HMM. Επιπλέον τα αντικείμενα αυτής της κλάσης συνθέτουν τις χειρονομίες. Οι μεταβλητές της κλάσης αυτής είναι οι εξής:

- **public int** index

  αποθηκεύεται ο δείκτης του δαχτύλου που δημιούργησε το σημείο

- **public float** x

  αποθηκεύεται η κανονικοποιημένη συντεταγμένη του σημείου στον άξονα X

- **public float** y

  αποθηκεύεται η κανονικοποιημένη συντεταγμένη του σημείου στον άξονα Ψ

- **public float** vx

αποθηκεύεται η ταχύτητα στον άξονα X όταν βρισκόμαστε στο σημείο

- **public float** vy

  αποθηκεύεται η ταχύτητα στον άξονα Υ όταν βρισκόμαστε στο σημείο

- **public float** pressure

  αποθηκεύεται η πίεση με την οποία δημιουργήθηκε το σημείο

- **public float** theta1

  αποθηκεύεται η γωνία του σημείου από το κέντρο της μάζας

- **public float** theta2

  αποθηκεύεται η γωνία του σημείου από το προηγούμενο σημείο

- **public float** location

  αποθηκεύεται η απόσταση του σημείου από το κέντρο μάζας

Ακολουθεί η παρουσίαση των μεθόδων της κλάσης αυτής.

5.3.1.1 public String toString()

η μέθοδος αυτή μετατρέπει ένα αντικείμενο της κλάσης σε συμβολοσειρά, έτσι ώστε να μπορεί να διαβαστεί και να αποθηκευτεί στη βάση δεδομένων.

```
public String toString() {
        String gesturePointString;
        gesturePointString = String.valueOf(index) + "," + String.
            valueOf(x) + "," +
                 String.valueOf(y) + "," + String.valueOf(vx) + "," +
                     String.valueOf(vy) + "," +
                 String.valueOf(pressure) + "," + String.valueOf(theta1)
                      + "," + String.valueOf(theta2) + ","
                + String.valueOf(location) + ";";

        return gesturePointString;
    }
```

5.3.1.2 public ObservationVector toObservationVector()

Η μέθοδος αυτή μετατρέπει ένα αντικείμενο της κλάσης GesturePoint σε αντικείμενο της κλάσης ObservationVector έτσι ώστε να μπορεί να χρησιμοποιηθεί από τη βιβλιοθήκη Jahmm.

```
public ObservationVector toObservationVector() {
        double[] obsVector;

        obsVector = new double[7];
        obsVector[0] = getIndex();
        obsVector[1] = getVx();
        obsVector[2] = getVy();
```

```
        obsVector[3] = getPressure();
        obsVector[4] = getTheta1();
        obsVector[5] = getTheta2();
        obsVector[6] = getLocation();


        ObservationVector observationVector = new ObservationVector(
            obsVector);
        return observationVector;
    }
```

5.3.1.3 Get/Set methods

Για την άμεση πρόσβαση και αλλαγή των μεταβλητών ενός αντικειμένου της κλάσης, έχουν υλοποιηθεί μέθοδοι get, set για καθεμία μεταβλητή και παρουσιάζονται συνολικά σε αυτή την ενότητα.

```
    public float getX() {
        return x;
    }
    public void setX(float x) {
        this.x = x;
    }
    public float getY() {
        return y;
    }
    public void setY(float y) {
        this.y = y;
    }
    public float getVx() {
        return vx;
    }
    public void setVx(float vx) {
        this.vx = vx;
    }
    public float getVy() {
        return vy;
    }
    public void setVy(float vy) {
        this.vy = vy;
    }
    public float getPressure() {
        return pressure;
    }
    public void setPressure(float pressure) {
        this.pressure = pressure;
    }
    public float getTheta1() {
        return theta1;
    }
    public void setTheta1(float theta1) {
        this.theta1 = theta1;
    }
    public float getTheta2() {
        return theta2;
```

```
    }
    public void setTheta2(float theta2) {
        this.theta2 = theta2;
    }
    public float getLocation() {
        return location;
    }
    public void setLocation(float location) {
        this.location = location;
    }
    public int getIndex() {
        return index;
    }
    public void setIndex(int index) {
        this.index = index;
    }
```

5.3.2 class Gesture

Η συγκεκριμένη κλάση αποτελεί επίσης μία από τις πιο σημαντικές της υλοποίησης, καθώς μοντελοποιεί τις χειρονομίες συνολικά. Οι μεταβλητές της κλάσης αυτής είναι οι εξής:

- `int id`

  αποθηκεύεται η μοναδική ταυτότητα κάθε χειρονομίας

- `int user_id`

  αποθηκεύεται η μοναδική ταυτότητα του χρήστη που δημιούργησε την χειρονομία

- `List<GesturePoint> gesturePointList = new ArrayList<>()`

  αποθηκεύεται η λίστα των σημείων που αποτελούν την χειρονομία

- `Date dateCaptured`

  αποθηκεύεται η ημερομηνία που συλλέχθηκε η χειρονομία

- `double score;`

  αποθηκεύεται το σκορ της χειρονομίας

Ακολουθεί η παρουσίαση των μεθόδων της κλάσης.

5.3.2.1 public float calculateCx()

Η μέθοδος αυτή υπολογίζει το κέντρο μάζας της χειρονομίας στον άξονα των X, χρησιμοποιώντας τις κανονικοποιημένες τιμές των συντεταγμένων στον άξονα των X.

```
for (GesturePoint gpoint : gesturePointList) {
            sumX += gpoint.x;
        }
        cx = sumX / gesturePointList.size();
```

5.3.2.2 public float calculateCy()

Η μέθοδος αυτή υπολογίζει το κέντρο μάζας της χειρονομίας στον άξονα των Ψ, χρησιμοποιώντας τις κανονικοποιημένες τιμές των συντεταγμένων στον άξονα των Ψ.

```
for (GesturePoint gpoint : gesturePointList) {
            sumY += gpoint.y;
        }
        cy = sumY / gesturePointList.size();
```

5.3.2.3 public void calculateLocation()

Η συγκεκριμένη μέθοδος υπολογίζει για όλα τα στοιχεία που συνθέτουν τη χειρονομία την απόσταση τους από το κέντρο μάζας της χειρονομίας και στη συνέχεια την κανονικοποιεί με βάση τη μέγιστη απόσταση όλων των σημείων από το κέντρο μάζας.

```
public void calculateLocation() {
float max = 0;
        float cx = calculateCx();
        float cy = calculateCy();
        for (GesturePoint gpoint : gesturePointList) {
            gpoint.location = (float) Math.sqrt(Math.pow((gpoint.x - cx
                ), 2) + Math.pow((gpoint.y - cy), 2));
            if (gpoint.location > max)
                max = gpoint.location;
        }
        //normalize location
        for (GesturePoint gpoint : gesturePointList) {
            gpoint.location = gpoint.location / max;
        }
}
```

5.3.2.4 public void calculateTheta1()

Η μέθοδος υπολογίζει για όλα τα σημεία που συνθέτουν το αντικείμενο της κλάσης την γωνία του από το κέντρο μάζας, κάνοντας χρήση της συνάρτησης $Math.atan$ που υπολογίζει την αντίστροφη εφαπτομένης, δηλαδή τη γωνία της δοσμένης τιμής εφαπτομένης. Σε περίπτωση που το σημείο ταυτίζεται με το κέντρο μάζας, η γωνία τίθεται ίση με 0.

```
public void calculateTheta1() {

        float cx = calculateCx();
        float cy = calculateCy();
        for (GesturePoint gpoint : gesturePointList) {
            if (cx != gpoint.x && cy != gpoint.y) {
                gpoint.theta1 = (float) Math.atan((gpoint.y - cy) / (
                    gpoint.x - cx));
            } else gpoint.theta1 = 0;
        }
    }
```

5.3.2.5 public void calculateTheta2()

Η μέθοδος υπολογίζει για όλα τα σημεία που συνθέτουν το αντικείμενο της κλάσης την γωνία του από το προηγούμενο σημείο, κάνοντας χρήση της συνάρτησης $Math.atan$ που υπολογίζει την αντίστροφη εφαπτομένης, δηλαδή τη γωνία της δοσμένης τιμής εφαπτομένης. Σε περίπτωση που το σημείο ταυτίζεται με το προηγούμενο σημείο, η γωνία τίθεται ίση με 0.

```
public void calculateTheta2() {
    for (int i = 1; i < gesturePointList.size(); i++) {
        if (gesturePointList.get(i).x != gesturePointList.get(i -
            1).x && gesturePointList.get(i).y != gesturePointList.
            get(i - 1).y) {
             gesturePointList.get(i).theta2 = (float) Math.atan(((
                 gesturePointList.get(i).y - gesturePointList.get(i
                 - 1).y) /
                      (gesturePointList.get(i).x - gesturePointList.
                          get(i - 1).x)));
        } else gesturePointList.get(i).theta2 = 0;
    }
}
```

5.3.2.6 public String toString()

Η μέθοδος μετατρέπει τη χειρονομία σε συμβολοσειρά, για την απεικόνιση και αποθήκευση της στη βάση δεδομένων.

```
public String toString() {
    String gestureString = "";
    for (GesturePoint gpoint : this.gesturePointList) {
        gestureString += gpoint.toString() + "\n";
    }

    return gestureString;
}
```

5.3.2.7 public List<ObservationVector> toListObservationVector()

Η μέθοδος αυτή είναι αρκετά σημαντική, καθώς μετατρέπει τη χειρονομία σε λίστα διανυσμάτων παρατήρησης, δηλαδή σε λίστα από ObservationVector η οποία θα χρησιμοποιηθεί από τις συναρτήσεις της βιβλιοθήκης Jahmm για την εκπαίδευση του μοντέλου.

```
public List<ObservationVector> toListObservationVector() {
    List<ObservationVector> obsList = new ArrayList<>();

    for (GesturePoint gpoint : this.gesturePointList) {
        obsList.add(gpoint.toObservationVector());
    }

    return obsList;

}
```

5.3.2.8 Get/Set methods

Για την πρόσβαση στα στοιχεία της κλάσης, καθώς και την τροποποίηση τους έχουν υλοποιηθεί η παρακάτω μέθοδοι:

```
public Date getDateCaptured() {
    return dateCaptured;
}
public void setDateCaptured(Date dateCaptured) {
    this.dateCaptured = dateCaptured;
}
public int getUser_id() {
    return user_id;
}
public void setUser_id(int user_id) {
    this.user_id = user_id;
}
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public double getScore() {
    return score;
}
public void setScore(double score) {
    this.score = score;
}
public List<GesturePoint> getGesturePointList() {
    return gesturePointList;
}
public void setGesturePointList(List<GesturePoint> gesturePointList
    ) {
    this.gesturePointList = gesturePointList;
}
```

5.3.3 class User

Η κλάση αυτή μοντελοποιεί τον χρήστη του συστήματος και οι μεταβλητές των αντικειμένων της κλάσης είναι οι εξής:

- **int** id

  αποθηκεύεται η μοναδική ταυτότητα κάθε χρήστη

- String userName

  αποθηκεύεται η συμβολοσειρά του ονόματος που διαλέγει ο κάθε χρήστης

- String phone_number

  αποθηκεύεται η συμβολοσειρά του τηλεφώνου κάθε χρήστη

- String hmm

αποθηκεύεται η συμβολοσειρά του HMM που αντιστοιχεί σε κάθε χρήστη.

Ακολουθεί η παρουσίαση των μεθόδων της κλάσης. Η κλάση αυτή έχει μεθόδους για την πρόσβαση και αλλαγή των στοιχείων κάθε χρήστη καθώς δεν είναι απαραίτητος κάποιος υπολογισμός για τα στοιχεία αυτής της κλάσης.

5.3.3.1 Get/Set methods

Παρατίθενται οι μέθοδοι για πρόσβαση και αλλαγή των μεταβλητών ενός αντικειμένου της κλάσης.

```
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getHmm() {
    return hmm;
}
public void setHmm(String hmm) {
    this.hmm = hmm;
}
public String getUserName() {
    return userName;
}
public void setUserName(String userName) {
    this.userName = userName;
}
public String getPhone_number() {
    return phone_number;
}
public void setPhone_number(String phone_number) {
    this.phone_number = phone_number;
}
```

5.3.4 class MainActivity

Η κλάση αυτή υλοποιεί την πρώτη οθόνη που εμφανίζεται στο χρήστη όταν ξεκινά την εφαρμογή. Η κλάση επεκτείνει την κλάση ActionBarActivity η οποία είναι η κλάση βάσης για όλες τις δραστηριότητες σε περιβάλλον Android που χρησιμοποιούν χαρακτηριστικά της μπάρας εργασιών. Επιπλέον υλοποιεί και την κλάση View.OnClickListener η οποία είναι μια διαπροσωπεία για να καλεστεί μία επανάκληση όταν γίνει ένα κλικ σε μία προβολή. Οι μεταβλητές των αντικειμένων αυτής της κλάσης είναι:

- Editext name

  πεδίο κειμένου στο οποίο ο χρήστης προσθέτει το όνομά του

- Editext phone

  πεδίο κειμένου στο οποίο ο χρήστης προσθέτει το τηλέφωνο του

Ακολουθεί η παρουσίαση των μεθόδων της κλάσης

5.3.4.1 protected void onCreate(Bundle savedInstanceState)

Η μέθοδος καλείται όταν η δραστηριότητα ξεκινάει και χρησιμοποιείται για αρχικοποίηση μεταβλητών. Ακολουθεί η περιγραφή των λειτουργιών της μεθόδου:

Θέτουμε τη διαρρύθμιση της δραστηριότητας

```
setContentView(R.layout.activity_main_form);
```

Η διαρρύθμιση ορίζεται από ένα xml αρχείο το οποίο είναι το εξής:

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ScrollView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:scrollbars="vertical">>

    <LinearLayout
        android:layout_width="fill_parent"
        android:orientation="vertical"
        android:layout_height="fill_parent">

        <TextView
            android:id="@+id/TextViewTitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/formDescription"
            android:textSize="10pt">
        </TextView>


        <EditText
            android:id="@+id/EditTextName"
            android:layout_height="wrap_content"
            android:hint="@string/userName"
            android:inputType="textPersonName"
            android:layout_width="fill_parent">
        </EditText>

        <EditText
            android:id="@+id/EditTextPhone"
            android:inputType="phone"
            android:hint="@string/phone"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <Button
            android:id="@+id/ButtonSendFeedback"
            android:layout_height="wrap_content"
            android:text="@string/okButton"
            android:onClick="onClick"
            android:layout_width="fill_parent">
        </Button>

</LinearLayout>
</ScrollView>
```

Στη συνέχεια θέτουμε τον τίτλο της δραστηριότητας και αρχικοποιούμε τις μεταβλητές πεδίων κειμένου.

```
setTitle("Login Form");
name = (EditText) this.findViewById(R.id.EditTextName);
phone = (EditText) this.findViewById(R.id.EditTextPhone);
```

5.3.4.2 public void onClick(View v)

Η μέθοδος αυτή υλοποιεί τη συμπεριφορά της εφαρμογής όταν στην προβολή στην οποία βρισκόμαστε πατηθεί το κουμπί που έχει σχεδιαστεί.

Αρχικά δημιουργούμε ένα νέο αντικείμενο χρήστη με βάση τα στοιχεία που δεχτήκαμε από τη φόρμα και στη συνέχεια βλέπουμε αν υπάρχει στη βάση δεδομένων χρήστης με τα στοιχεία που μόλις δεχτήκαμε. Αν υπάρχει ο χρήστης στη βάση, οδηγείται στη δραστηριότητα ελέγχου του HMM του με νέες χειρονομίες, ενώ αν δεν υπάρχει οδηγείται στη δραστηριότητα εκπαίδευσης του HMM που θα του αντιστοιχηθεί. Για ευκολία στην εξαγωγή της βάσης έχει προστεθεί και μία τρίτη επιλογή κατά την οποία η βάση δεδομένων εξάγεται σε ένα αρχείο στην κάρτα μνήμης του τηλεφώνου, εφόσον βάλουμε για username το όνομα admin.

Και στις δύο δραστηριότητες στις οποίες μεταφερόμαστε παιρνάμε σαν παραμέτρους και τα στοιχεία του χρήστη.

```
public void onClick(View v) {


        //create user and check if  exists in the db
        User newUser = new User();
        newUser.setUserName(name.getText().toString().toUpperCase());
        newUser.setPhone_number(phone.getText().toString());
        DatabaseHelper db = new DatabaseHelper(getApplicationContext())
            ;
        if (newUser.getUserName().equals("ADMIN")) {
            db.exportDB();
            Log.d("DB EXPORTED", "DB EXPORTED");

        } else if (!db.userExists(newUser)) { //add the user in the db


            long id = db.createUser(newUser);
            newUser.setId((int) id);


            //go to TrainActivity

            Intent i = new Intent(getApplicationContext(),
                TrainActivity.class);
            i.putExtra("user", newUser.getId());
            startActivity(i);
            Toast toast = Toast.makeText(getApplicationContext(),
                    "Press Cancel for initialization and then Enter the
                        Gesture for training " +
                            "the model and " +
                            "press ok",
```

```
                    Toast.LENGTH_LONG);
            toast.show();
            setContentView(R.layout.activity_train);
        } else {
            //user exists so we go to probability calculation
            Intent i = new Intent(getApplicationContext(), TestActivity
                .class);
            i.putExtra("user", db.getUserIdFromName(newUser));
            startActivity(i);
            setContentView(R.layout.activity_train);

            finish();
        }
    }
```

5.3.5 class TrainActivity

Η κλάση αυτή υλοποιεί οθόνη που εμφανίζεται στο νέο χρήστη όπου θα πρέπει να εισάγει τις χειρονομίες για αν εκπαιδευτεί το HMM του. Η κλάση επεκτείνει την κλάση ActionBarActivity η οποία είναι η κλάση βάσης για όλες τις δραστηριότητες σε περιβάλλον Android που χρησιμοποιούν χαρακτηριστικά της μπάρας εργασιών.

Οι μεταβλητές της κλάσης αυτής είναι

- `User newUser`

  ο νέος χρήστης που μόλις εισήλθε στο σύστημα

- `TextView textView`

  πεδίο κειμένου στο οποίο θα φαίνονται τα σημείο που ακουμπά ο χρήστης

- `ImageView imageView`

  η προβολή που δέχεται το background της δραστηριότητας και πάνω στην οποία σχεδιάζει ο χρήστης

- `Bitmap bitmap`

  χρησιμοποιείται για την σχεδίαση και αποθήκευση της χειρονομίας του χρήστη

- `Canvas canvas`

  ο καμβάς πάνω στον οποίο σχεδιάζει ο χρήστης

- `Paint paint`

  το χρώμα με το οποίο σχεδιάζονται οι χειρονομίες

- **float** `downx, downy`

  οι συντεταγμένες που προσγειώνεται αρχικά το δάχτυλο του χρήστη

- **float** `vx,vy`

οι ταχύτητες στους άξονες κατά την κίνηση

- **float** movex, movey

  οι συντεταγμένες κίνησης

- **float** canvasHeight, canvasWidth

  οι διαστάσεις του καμβά σχεδίασης

- **int** index

  ο δείκτης κάθε δαχτύλου που ακουμπά την οθόνη

- **boolean** firstPointerFlag

  λογική μεταβλητή που δείχνει εάν το δάχτυλο που ακούμπησε την οθόνη είναι το πρώτο που την ακουμπά

- **int** maxTrain

  οι χειρονομίες που απαιτούμε από το χρήστη για την εκπαίδευση του μοντέλου

- **int** countTrain

  μετρητής των χειρονομιών που μας δίνει ο χρήστης

- **int** nbTrains

  το πλήθος των φορών με τις οποίες εκπαιδεύουμε το HMM με μία χειρονομία του χρήστη.

- List<GesturePoint> gesture = **new** ArrayList<>();

  η λίστα των σημείων που καταγράφονται

- GesturePoint gesturePoint

  το σημείο που καταγράφεται κάθε στιγμή

- Gesture gestureFinal

  η τελική χειρονομία του χρήστη

- Hmm<ObservationVector> hmm

  το HMM του χρήστη

- List<List<ObservationVector>> gestureTrain = **new** ArrayList<>();

  η τελική χειρονομία του χρήστη μετασχηματισμένη σε μορφή αναγνωρίσιμη από τη βιβλιοθήκη Jahmm

- BaumWelchScaledLearner bwl

αντικείμενο της κλάσης BaumWelchScaledLearner το οποίο θα χρησιμοποιηθεί για την εκπαίδευση του μοντέλου

- `private VelocityTracker mVelocityTracker`

  αντικείμενο της κλάσης VelocityTracker το οποίο θα μετράει την ταχύτητα σε κάθε σημείο που καταγράφεται.

Ακολουθεί η παρουσίαση των μεθόδων της κλάσης

5.3.5.1 protected void onCreate(Bundle savedInstanceState)

Η μέθοδος καλείται όταν η δραστηριότητα ξεκινάει και χρησιμοποιείται για αρχικοποίηση μεταβλητών. Ακολουθεί η περιγραφή των λειτουργιών της μεθόδου:

Θέτουμε τη διαρρύθμιση της δραστηριότητας

```
setContentView(R.layout.activity_train);
```

όπως αυτή ορίζεται από το εξής xml αρχείο και φαίνεται στην εικόνα 4.3

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width
        ="fill_parent"
    android:layout_height="fill_parent" android:paddingLeft="@dimen/
        activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
        tools:context=".MainActivity"
    android:background="@drawable/bg">


    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/ImageViewID" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="(x,y)"
        android:id="@+id/textView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignTop="@+id/ImageViewID" />



    <Button
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="␣␣␣␣OK␣␣␣␣"
        android:id="@+id/buttonOk"
```

```
        android:layout_alignRight="@+id/ImageViewID"
        android:layout_alignEnd="@+id/ImageViewID"
        android:layout_alignParentBottom="true"
        android:onClick="ok" />

    <Button
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:id="@+id/buttonCancel"
        android:onClick="cancel"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />


</RelativeLayout>
```

Στη συνέχεια παίρνουμε τα στοιχεία που μας ήρθαν από την προηγούμενη δραστηριότητα και δημιουργούμε ένα χρήστη με βάση το id του χρήστη που μόλις ήρθε στο σύστημα. Επιπλέον αρχικοποιούμε τις μεταβλητές textView, imageView

```
Bundle extras = getIntent().getExtras();


        if (extras != null) {
            int temp = extras.getInt("user");
            Log.d("TEMP", String.valueOf(temp));
            newUser.setId(temp);
            Log.d("USER ID = ", String.valueOf(newUser.getId()));
        }

        imageView = (ImageView) this.findViewById(R.id.ImageViewID);
        textView = (TextView) this.findViewById(R.id.textView);
        textView.setText("Press Cancel to create canvas");
```

5.3.5.2 public void onStart()

Η μέθοδος καλείται αμέσως μετά την onCreate() και την χρησιμοποιούμε για την αρχικοποίηση υπόλοιπων μεταβλητών και την τοποθέτηση του touchListener στην προβολή.

```
bitmap = Bitmap.createBitmap((int) canvasWidth, (int) canvasHeight,
                Bitmap.Config.ARGB_8888);
canvas = new Canvas(bitmap);
textView.setText("(x,y)");
paint = new Paint();
paint.setColor(Color.GREEN);
paint.setStyle(Paint.Style.STROKE);
imageView.setImageBitmap(bitmap);
```

5.3.5.3 public boolean onTouch(View v, MotionEvent event)

Αυτή η μέθοδος υλοποιεί το κομμάτι καταγραφής της χειρονομίας του χρήστη και η λειτουργία της μπορεί να χωριστεί στις εξής περιπτώσεις:

1. MotionEvent.ACTION-DOWN: το συμβάν κατά το οποίο το πρώτο δάχτυλο του χρήστη ακουμπά την οθόνη. Στην περίπτωση αυτή δημιουργούμε μία νέα λίστα στην οποία θα αποθηκεύσουμε τα υπόλοιπα σημεία της χειρονομίας στη συνέχεια και καταγράφουμε τα μετρήσιμα μεγέθη, δηλαδή τις συντεταγμένες, τις ταχύτητες στους δύο άξονες και την πίεση. Στη συνέχεια ζωγραφίζουμε το σημείο.

2. MotionEvent.ACTION-POINTER-DOWN: το συμβάν μοντελοποιεί την περίπτωση που ο χρήστης ακουμπά επιπλέον δάχτυλα στην οθόνη. Κάθε νέο δάχτυλο αποκτά μία ταυτότητα και όλα τα μετρήσιμα μεγέθη καταγράφονται και προστίθενται στη λίστα της χειρονομίας όπως και προηγουμένως. Στη συνέχεια ζωγραφίζουμε το σημείο.

3. MotionEvent.ACTION-MOVE: το συμβάν αυτό ενεργοποιείται όταν κάποιο από τα δάχτυλα κουνιέται. Για κάθε δάχτυλο που κουνιέται καταγράφεται το σημείο και προστίθεται στη λίστα της χειρονομίας και στη συνέχεια σχεδιάζεται στον καμβά.

```
public boolean onTouch(View v, MotionEvent event) {
            int action = MotionEventCompat.getActionMasked(event);


            switch (action) {
                case MotionEvent.ACTION_DOWN: //first pointer down
                    if (firstPointerFlag) {
                        gesture = new ArrayList<>();
                    }
                    firstPointerFlag = false;


                    index = MotionEventCompat.getActionIndex(event); //
                        index will be 0
                    downx = MotionEventCompat.getX(event, index);
                    downy = MotionEventCompat.getY(event, index);

                    if (mVelocityTracker == null) {
                        // Retrieve a new VelocityTracker object to
                            watch the velocity of a motion.
                        mVelocityTracker = VelocityTracker.obtain();
                    } else {
                        // Reset the velocity tracker back to its
                            initial state.
                        mVelocityTracker.clear();
                    }
                    // Add a user's movement to the tracker.
                    mVelocityTracker.addMovement(event);
                    mVelocityTracker.computeCurrentVelocity(1000);
```

```
            vx = VelocityTrackerCompat.getXVelocity(
                mVelocityTracker, index);
            vy = VelocityTrackerCompat.getYVelocity(
                mVelocityTracker, index);
            pressure = event.getPressure(index);

            gesturePoint = new GesturePoint(index, downx /
                canvasWidth, downy / canvasHeight, vx, vy,
                pressure);
            //  Log.d("GesturePoint added", gesturePoint.
                toString());

            gesture.add(gesturePoint);
            imageView.invalidate();
            // canvas.drawPoint(downx, downy, paint);

            textView.setText("x=" + String.valueOf(downx) + ",␣
                y=" + String.valueOf(downy));
            break;

        case MotionEvent.ACTION_POINTER_DOWN: //another pointer
            added


            index = MotionEventCompat.getActionIndex(event);
            downx = MotionEventCompat.getX(event, index);
            downy = MotionEventCompat.getY(event, index);
            if (mVelocityTracker == null) {
                // Retrieve a new VelocityTracker object to
                    watch the velocity of a motion.
                mVelocityTracker = VelocityTracker.obtain();
            } else {
                // Reset the velocity tracker back to its
                    initial state.
                mVelocityTracker.clear();
            }
            // Add a user's movement to the tracker.
            mVelocityTracker.addMovement(event);
            mVelocityTracker.computeCurrentVelocity(500);
            vx = VelocityTrackerCompat.getXVelocity(
                mVelocityTracker, index);
            vy = VelocityTrackerCompat.getYVelocity(
                mVelocityTracker, index);
            pressure = event.getPressure(index);


            gesturePoint = new GesturePoint(index, downx /
                canvasWidth, downy / canvasHeight, vx, vy,
                pressure);
            // Log.d("GesturePoint added", gesturePoint.
                toString());
            gesture.add(gesturePoint);
            imageView.invalidate();

            // canvas.drawCircle(downx, downy, 6, paint);
```

```
        //canvas.drawPoint(downx, downy, paint);

    case MotionEvent.ACTION_UP:
        index = MotionEventCompat.getActionIndex(event);


    case MotionEvent.ACTION_MOVE:

        if (mVelocityTracker == null) {
            // Retrieve a new VelocityTracker object to
                watch the velocity of a motion.
            mVelocityTracker = VelocityTracker.obtain();
        } else {
            // Reset the velocity tracker back to its
                initial state.
            mVelocityTracker.clear();
        }
        // Add a user's movement to the tracker.
        mVelocityTracker.addMovement(event);

        int count = event.getPointerCount();
        for (int i = 0; i < count; i++) {
            mVelocityTracker.computeCurrentVelocity(500);


            movex = MotionEventCompat.getX(event, i);
            movey = MotionEventCompat.getY(event, i);


            vx = VelocityTrackerCompat.getXVelocity(
                mVelocityTracker, i);
            vy = VelocityTrackerCompat.getYVelocity(
                mVelocityTracker, i);
            pressure = event.getPressure(i);

            gesturePoint = new GesturePoint(i, movex /
                canvasWidth, movey / canvasHeight, vx, vy,
                pressure); //normalized x,y values

            gesture.add(gesturePoint);
            imageView.invalidate();

            // Log.d("GesturePoint Added", gesturePoint.
                toString());


            float prevx = 0, prevy = 0;
            for (int k = gesture.size(); k > 0; k--) {
                prevx = canvasWidth * gesture.get(k - 2).x;
                prevy = canvasHeight * gesture.get(k - 2).y
                    ;
                if (gesture.get(k - 2).index == i)
                    break;
            }
```

```
                    //Log.d("PREV", String.valueOf(prevx)+" "+
                        String.valueOf(prevy));
                    //canvas.drawCircle(movex, movey, 6, paint);
                    //canvas.drawPoint(downx, downy, paint);

                    canvas.drawLine(prevx, prevy, movex, movey,
                        paint2);
                    textView.setText("x=" + String.valueOf(movex) +
                        ", y=" + String.valueOf(movey));
                }

                mVelocityTracker.clear();
            case MotionEvent.ACTION_CANCEL:
                mVelocityTracker.clear();
                break;
            default:
                break;
        }


        return true;
    }
```

5.3.5.4 public void onDraw(Canvas canvas)

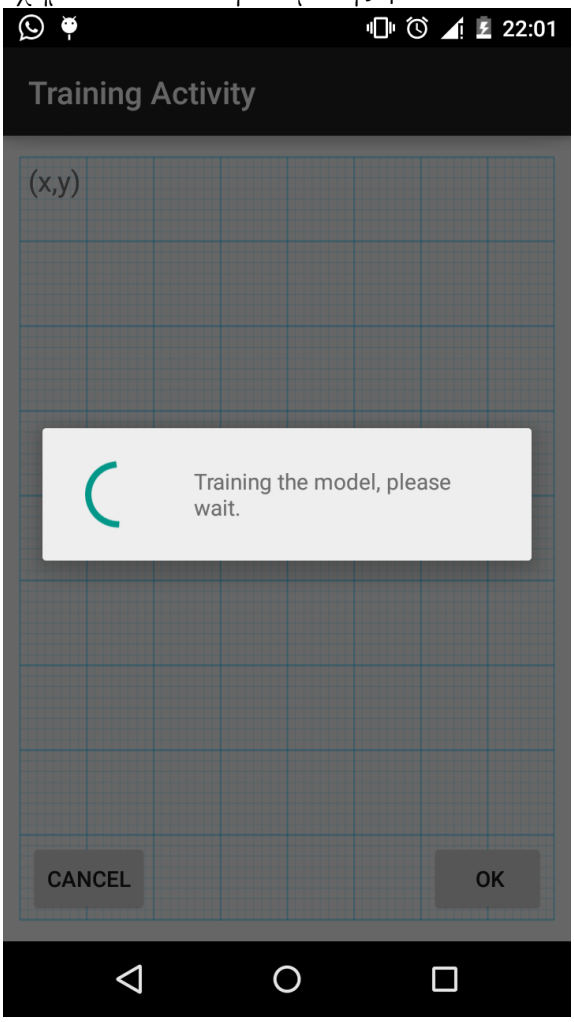
Η μέθοδος αυτή καλείται για την σχεδίαση πάνω στον καμβά

```
public void onDraw(Canvas canvas) {
        for (GesturePoint point : gesture) {
            canvas.drawCircle(point.getX() * canvasWidth, point.
                getY() * canvasHeight, 5, paint);
            // Log.d("ONDRAW","ONDRAW CALLED");
        }
    }
```

5.3.5.5 public void ok(View view)

Η μέθοδος αυτή καλείται όταν ο χρήστης είναι έτοιμος να καταχωρήσει την χειρονομία του στο σύστημα και είναι υπεύθυνη για αρκετές εργασίες. Υπολογίζει για κάθε σημείο της χειρονομίας τη γωνία του και την απόστασή του από το κέντρο μάζας, τη γωνία του από το προηγούμενο σημείο και στη συνέχεια αποθηκεύει την πλήρη χειρονομία στη βάση δεδομένων. Στη συνέχεια μετατρέπει τη χειρονομία σε μορφή που αναγνωρίζεται από τη βιβλιοθήκη Jahmm. Εφόσον είναι η πρώτη χειρονομία που εισάγει ο χρήστης η μέθοδος αυτή δημιουργεί μία πρώτη εκτίμηση του μοντέλου χρησιμοποιώντας τον αλγόριθμο Kmeans, έτσι ώστε να είμαστε σίγουροι ότι το μοντέλο θα συγκλίνει μετά την εκπαίδευσή του. Στη συνέχεια δημιουργεί ένα νήμα με τη βοήθεια της κλάσης AsyncTask έτσι ώστε να αναλάβει τη βαριά υπολογιστικά διαδικασία της εκπαίδευσης του μοντέλου. Κατά τη διάρκεια της εκπαίδευσης ο χρήστης βλέπει ένα σήμα ότι το σύστημα είναι απασχολημένο 5.1 Εφόσον ο χρήστης συμπληρώσει τις 10 χειρονομίες που απαιτούνται, αποθηκεύεται στη βάση δεδομένων το μοντέλο που δημιούργησε και οδηγείται στην αρχική οθόνη για να εισέλθει ξανά στο σύστημα με σκοπό να ελέγξει τις χειρονομίες του με το μοντέλο που δημιούργησε.

Σχήμα 5.1: Οθόνη αναμονής για εκπαίδευση



```
public void ok(View view) {


        Log.d("Ok pressed", "ok");


        gestureFinal = new Gesture(gesture);


        gestureFinal.calculateTheta1();
        gestureFinal.calculateTheta2();
        gestureFinal.calculateLocation();

        Log.d("GESTURECAPTURED", gestureFinal.toString());
        DatabaseHelper db = new DatabaseHelper(getApplicationContext())
            ;
        int gest_id = (int) db.createGesture(gestureFinal, newUser);
        if (countTrain <= maxTrain) {

            /*
write the gesture to the database
 */


            gestureTrain.add(gestureFinal.toListObservationVector());
            TrainingTask task = new TrainingTask(TrainActivity.this);
            if (countTrain == 0) {
                hmm = buildInitHmm();

                task.execute();


            } else {
                task.execute();


            }
            countTrain++;
            this.onStart();

        } else {
            if (countTrain <= maxTrain + 1) {
                HmmWriter hmmWriter = new HmmWriter();
                HmmBinaryWriter hmmBinaryWriter = new HmmBinaryWriter()
                    ;
                StringWriter stringWriter = new StringWriter();
                ByteArrayOutputStream byteArrayOutputStream = new
                    ByteArrayOutputStream();

                try {
                    hmmWriter.write(stringWriter, new
                        OpdfMultiGaussianWriter(), hmm);
                    hmmBinaryWriter.write(byteArrayOutputStream, hmm);
                } catch (IOException e) {
                    e.printStackTrace();
```

```
                }
                byte[] hmmByte = byteArrayOutputStream.toByteArray();
                db.updateUserString(newUser, stringWriter.toString());
                db.updateUserBytes(newUser, hmmByte);


                Log.d("USER'S HMM", newUser.getHmm());
                countTrain++;
            }


            Intent i = new Intent(getApplicationContext(), MainActivity
                .class);


            startActivity(i);
            setContentView(R.layout.activity_main_form);


        }
    }
```

5.3.5.6 public void cancel(View view)

Η μέθοδος αυτή καλείται όταν ο χρήστης δεν θέλει να υποβάλει την χειρονομία του και καθαρίζει όλες τις μεταβλητές και επιστρέφει στην αρχική κατάσταση, αναμένοντας την χειρονομία του χρήστη.

```
public void cancel(View view) {
        Log.d("cancel pressed", "cancel");
        gesture.clear();
        this.onStart();

    }
```

5.3.5.7 private Hmm<ObservationVector> buildInitHmm()

Η μέθοδος αυτή είναι υπεύθυνη για τη δημιουργία μιας πρώτης εκτίμησης του μοντέλου. Χρησιμοποιώντας τη συνάρτηση KMeansLearner της βιβλιοθήκης Jahmm, δημιουργούμε μία πρώτη εκτίμηση για τις καταστάσεις του μοντέλου, έτσι ώστε να είμαστε βέβαιοι ότι το μοντέλο θα συγκλίνει. Η λειτουργία της είναι η εξής:

Ο αλγόριθμος τοποθετεί τα σημεία σε συστάδες χρησιμοποιώντας την κλάση KMeansCalculator. Η συγκεκριμένη κλάση δέχεται στοιχεία σαν είσοδο και παράγει συστάδες από αυτά τα στοιχεία σαν έξοδο. Το παραγόμενο HMM μπορεί να χρησιμοποιηθεί σαν μία καλή αφετηρία για τον αλγόριθμο Baum-Welch.

```
private Hmm<ObservationVector> buildInitHmm() {
        KMeansLearner<ObservationVector> kml = new KMeansLearner<>(
            nbStates, new
                OpdfMultiGaussianFactory(7), gestureTrain);

        hmm = kml.learn();
```

```
        return hmm;
    }
```

5.3.6 class TrainingTask

Η κλάση αυτή επεκτείνει την κλάση AsyncTask που περιγράφηκε προηγουμένως και χρησιμοποιείται για να γίνει η βαριά υπολογιστικά διαδικασία της εκπαίδευσης του μοντέλου, καθώς και για την εμφάνιση του μηνύματος αναμονής 5.1. Η μεταβλητή της κλάσης είναι

```
ProgressDialog dialog
```

η οποία είναι το μήνυμα αναμονής που θα εμφανίζεται στο χρήστη όσο εκπαιδεύεται το μοντέλο.

Ακολουθεί η περιγραφή των μεθόδων της κλάσης.

5.3.6.1 protected void onPreExecute()

Η μέθοδος καλείται πρίν την εκτέλεση του νήματος παρασκηνίου που θα κάνει τον υπολογισμό, και είναι υπεύθυνη για την εμφάνιση του μηνύματος αναμονής.

```
protected void onPreExecute() {

        dialog.setMessage("Training the model, please wait.");
        dialog.show();
    }
```

5.3.6.2 protected Void doInBackground(Void... params)

Η μέθοδος είναι υπεύθυνη για την εκτέλεση της εκπαίδευσης του μοντέλου και τρέχει σε ένα νήμα παρασκηνίου. Κάνει χρήση του αλγόριθμου Baum-Welch , πάνω στην αρχική εκτίμηση του μοντέλου που έχουμε πάρει από τον αλγόριθμο KMeans, μέχρι να παραχθεί ένα σταθερό μοντέλο για τον χρήστη.

```
protected Void doInBackground(Void... params) {
        try {
            for (int i = 0; i < nbTrains; i++) {
                hmm = bwl.iterate(hmm, gestureTrain);

            }
            Log.d("HMM_ASYNC", hmm.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

5.3.6.3 protected void onPostExecute(Void result)

Η μέθοδος καλείται μετά το τέλος του υπολογισμού στο νήμα παρασκηνίου και σβήνει το μήνυμα αναμονής. Επιπλέον εμφανίζει και οδηγίες στο χρήστη για την εισαγωγή νέας χειρονομίας.

```
protected void onPostExecute(Void result) {
            //remove dialog
            if (dialog.isShowing())
                dialog.dismiss();
            Toast toast = Toast.makeText(getApplicationContext(),
                    "Enter the Gesture for training the model and press
                        ok",
                    Toast.LENGTH_LONG);
            toast.show();
           }
```

5.3.7 class TestActivity

Η κλάση αυτή υλοποιεί οθόνη που εμφανίζεται στο νέο χρήστη όπου θα πρέπει να εισάγει τις χειρονομίες για να εξεταστεί το σκορ τους με βάση το HMM του. Η κλάση επεκτείνει την κλάση ActionBarActivity η οποία είναι η κλάση βάσης για όλες τις δραστηριότητες σε περιβάλλον Android που χρησιμοποιούν χαρακτηριστικά της μπάρας εργασιών. Η κλάση αυτή είναι όμοια σε εμφάνιση με την κλάση για TrainActivity ως προς την εμφάνιση και την καταγραφή των χειρονομιών. Επομένως έχει ίδια διαμόρφωση και χρησιμοποιεί επίσης αντικείμενο της κλάσης View.OnTouchListener με τον ίδιο τρόπο όπως και η κλάση TrainActivity. Η μοναδική διαφορά της συγκεκριμένης κλάσης με την κλάση για την εκπαίδευση του μοντέλου, είναι η συμπεριφορά της αφού ο χρήστης επιβεβαιώσει την χειρονομία του πατώντας ok.

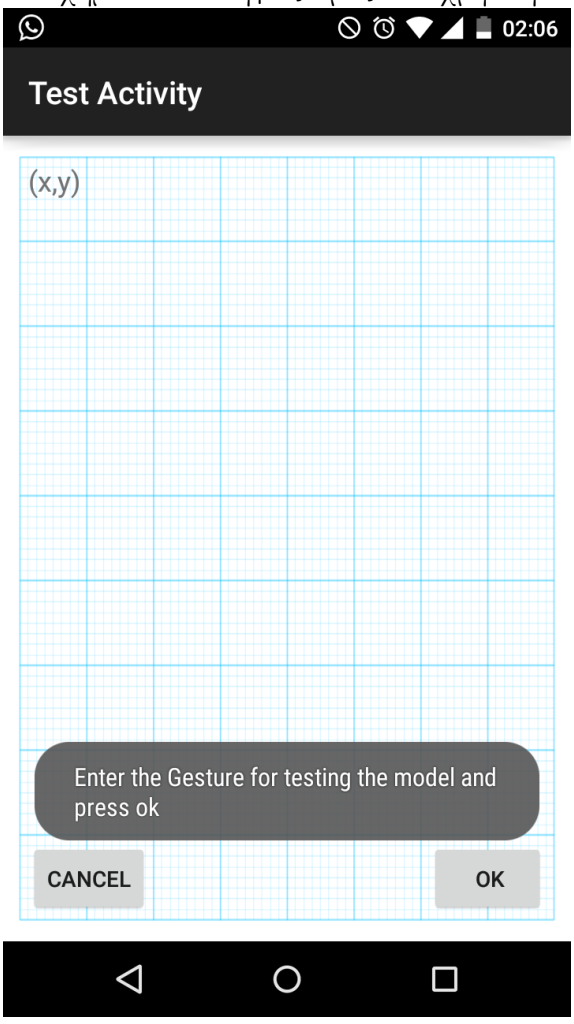
Πατώντας το ok ο χρήστης, ενεργοποιείται η παρακάτω μέθοδος

5.3.7.1 public void ok(View view)

Στη μέθοδο αυτή υπολογίζονται τα στοιχεία της χειρονομίας που δεν καταγράφονται αλλά υπολογίζονται μετά την καταγραφή. Επιπλέον το HMM του χρήστη που μπήκε στο σύστημα φέρνεται από τη βάση και με βάση αυτό υπολογίζεται το σκορ της χειρονομίας που καταγράφηκε. Στη συνέχεια το σκορ καταχωρείται μαζί με την χειρονομία στη βάση δεδομένων. Το σκορ υπολογίζεται χρησιμοποιώντας τη μέθοδο lnProbability της κλάσης HMM η οποία χρησιμοποιεί αντικείμενα της κλάσης ForwardBackwardCalculator που υλοποιούν τον αλγόριθμο για την εύρεση της πιθανότητας την οποία χρησιμοποιούμε ως σκορ. Επιπλέον καθ' όλη τη διάρκεια της δραστηριότητας ο χρήστης βλέπει οδηγίες για το επόμενο βήμα 5.2

```
public void ok(View view) {


        Log.d("Ok pressed", "ok");

        //time we started the testing
        //long timeStart = System.currentTimeMillis();
        gestureFinal = new Gesture(gesture);


        gestureFinal.calculateTheta1();
        gestureFinal.calculateTheta2();
        gestureFinal.calculateLocation();
```

Σχήμα 5.2: Οδηγίες προς τον χρήστη



```
        if (countTest < maxTest) {

            /*
write the gesture to the database
 */
            DatabaseHelper db = new DatabaseHelper(
                getApplicationContext());


            int gest_id = (int) db.createGesture(gestureFinal, newUser)
                ;
            gestureFinal.setId(gest_id);


            //find the user's hmm from the database

            String hmmString = db.getHMMFromUser(newUser);
            hmmString = hmmString.replaceAll("[,]", "");

            //writeToFile(hmmString,"HMM.txt");


            Log.d("HMM_RETURNED", hmmString);
            try {
                /* hmm = HmmReader.read(new FileReader( new File(
                    getExternalFilesDir(null), "HMM.txt")),
                            new OpdfMultiGaussianReader());*/
                 byte[] hmmByte = db.getHmmFromUser(newUser);
                 hmm = (Hmm<ObservationVector>) HmmBinaryReader.read(new
                       ByteArrayInputStream(hmmByte));


            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                db.close();
            }

            Log.d("HMM_READ", hmm.toString());


            double lnProb;

            lnProb = hmm.lnProbability(gestureFinal.
                toListObservationVector());

            gestureFinal.setScore(lnProb);
            Log.d("GESTURE_SCORE", String.valueOf(gestureFinal.getScore
                ()));
            Toast toast = Toast.makeText(getApplicationContext(),
                    "The_score_of_the_gesture_is:_"+ String.valueOf(
                        gestureFinal.getScore()),
                     Toast.LENGTH_LONG);
```

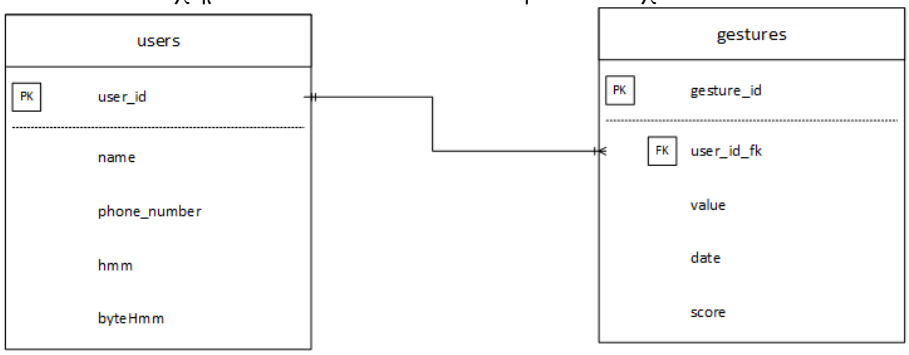
```
            toast.show();

            db.updateGesture(gest_id, lnProb);

            this.onStart();

        }
    }
```

5.3.8 class DatabaseHelper

Η κλάση αυτή επεκτείνει την κλάση SQLiteOpenHelper η οποία είναι μια βοηθητική κλάση για την διαχείριση μιας βάσης δεδομένων. Η κλάση υλοποιεί μεθόδους για τη δημιουργία και την αναβάθμιση της βάσης δεδομένων. Η σχεσιακή βάση δεδομένων που δημιουργήθηκε για τις ανάγκες της εφαρμογής αποτελείται από δύο πίνακες, ένα για τους χρήστες και ένα για τις χειρονομίες. Το μοντέλο οντοτήτων-συσχετίσεων της βάσης φαίνεται στο 5.3 Το αρχείο της βάσης που κατασκευάστηκε είναι το εξής:

```
CREATE TABLE users ( user_id INTEGER PRIMARY KEY NOT NULL, name TEXT ,
    phone_number TEXT,hmm TEXT, byteHmm BLOB);
CREATE TABLE gestures (gesture_id INTEGER PRIMARY KEY NOT NULL,
    user_id_fk INTEGER REFERENCES users (user_id) on DELETE CASCADE,
    value TEXT, date DATE, score REAL );
```

Σχήμα 5.3: Μοντέλο οντοτήτων-συσχετίσεων



Οι παράμετροι της κλάσεις έχουν να κάνουν με τα ονόματα των πινάκων που δημιουργούνται και με τα ονόματα των στηλών τους.

```
private static final int DATABASE_VERSION = 2;

    // Database Name
    private static final String DATABASE_NAME = "GestureUsers.db";

    //Users table name
    private static final String TABLE_USERS = "users";
```

```
// Users Table Columns names
private static final String USER_ID = "user_id";
private static final String USER_NAME = "name";
private static final String USER_PH_NO = "phone_number";
private static final String USER_HMM = "hmm";
private static final String USER_HMM_BYTES = "byteHmm";
//private static final String USER_EMAIL = "email";

//Gestures table name
private static final String TABLE_GESTURES = "gestures";

// Gestures Table Columns names
private static final String GESTURE_ID = "gesture_id";
private static final String USER_ID_FK = "user_id_fk";
private static final String GESTURE_VALUE = "value";
private static final String GESTURE_DATE = "date";
private static final String GESTURE_SCORE = "score";

//Create statements
private static final String CREATE_TABLE_USERS = "CREATE TABLE " +
    TABLE_USERS + " ( " + USER_ID + " INTEGER PRIMARY KEY NOT NULL,
    " + USER_NAME
        + " TEXT , " + USER_PH_NO + " TEXT, "
        + USER_HMM + " TEXT, " + USER_HMM_BYTES + " BLOB)";
private static final String CREATE_TABLE_GESTURES = "CREATE TABLE "
    + TABLE_GESTURES + " ( " + GESTURE_ID + " INTEGER PRIMARY KEY
    NOT NULL, " + USER_ID_FK
        + " INTEGER, " + GESTURE_VALUE + " TEXT, " + GESTURE_DATE +
            " DATE, " + GESTURE_SCORE + " REAL, FOREIGN KEY( " +
            USER_ID_FK + " ) REFERENCES " + TABLE_USERS + " ( " +
            USER_ID + " ) );
```

Οι μέθοδοι της κλάσης παρουσιάζονται παρακάτω:

5.3.8.1 public void onCreate(SQLiteDatabase db)

Η μέθοδος δημιουργεί αν δεν υπάρχουν ήδη τους πίνακες της βάσης.

```
public void onCreate(SQLiteDatabase db) {

    db.execSQL(CREATE_TABLE_USERS);
    db.execSQL(CREATE_TABLE_GESTURES);
}
```

5.3.8.2 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)

Η μέθοδος σε περίπτωση αναβάθμισης της βάσης δεδομένων σβήνει τους παλιούς πίνακες και ξαναφτιάχνει τη βάση.

```
        public void onUpgrade(SQLiteDatabase db, int oldVersion,
            int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_USERS);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_GESTURES);
```

```
        onCreate(db);
    }
```

5.3.8.3 public long createUser(User user)

Η μέθοδος αυτή δημιουργεί στη βάση ένα νέο χρήστη με τα στοιχεία του χρήστη που περνιέται σαν παράμετρος,

```
public long createUser(User user) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();

        values.put(USER_NAME, user.getUserName().toUpperCase());
        values.put(USER_PH_NO, user.getPhone_number());
       // values.put(USER_EMAIL, user.getEmail());

        //insert row
        long user_id = db.insert(TABLE_USERS, null, values);
        db.close();
        return user_id;
    }
```

5.3.8.4 public long createGesture(Gesture gesture,User user)

Η μέθοδος αυτή δημιουργεί στη βάση μία νέα χειρονομία που περνιέται σαν παράμετρος και ανήκει στον χρήστη που περνιέται σαν παράμετρος,

```
public long createGesture(Gesture gesture, User user) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();

        values.put(GESTURE_VALUE, gesture.getGesturePointList().
            toString());
        values.put(USER_ID_FK, user.getId());
        values.put(GESTURE_DATE, String.valueOf(gesture.getDateCaptured
            ()));

        //insert row
        long gesture_id = db.insert(TABLE_GESTURES, null, values);

        db.close();
        return gesture_id;
    }
```

5.3.8.5 public ArrayList<Gesture> getAllGestures(User user)

Η μέθοδος αυτή φέρνει από τη βάση όλες τις χειρονομίες που ανήκουν στον χρήστη που περνιέται σαν παράμετρος,

```
public ArrayList<Gesture> getAllGestures(User user) {
        ArrayList<Gesture> gestureList = new ArrayList<Gesture>();
        // Select All Query
        String selectQuery = "SELECT " + GESTURE_VALUE + " FROM " +
            TABLE_GESTURES +
```

```
                " WHERE " + USER_ID_FK + " =?";

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, new String[]{String.
            valueOf(user.getId())});

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                Gesture ges = new Gesture();
                ges.setId(Integer.parseInt(cursor.getString(0)));
                ges.setUser_id(cursor.getInt(1));
                //todo: parse the string into a gesture
                // ges.setGesturePointList(new ArrayList<GesturePoint>(
                    cursor.getString(2)));


                // Adding ges to list
                gestureList.add(ges);
            } while (cursor.moveToNext());
        }

        // return gesture list
        return gestureList;
    }
```

5.3.8.6 public boolean userExists(User user)

Η μέθοδος αυτή ελέγχει εάν ο χρήστης που περνιέται σαν παράμετρος υπάρχει στη βάση και επιστρέφει true αν υπάρχει και false εάν δεν υπάρχει

```
public boolean userExists(User user) {
        String countQuery = "SELECT * FROM " + TABLE_USERS + " tu WHERE
            tu." + USER_NAME + " =?";
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, new String[]{user.
            getUserName()});

        if (cursor != null && cursor.getCount() > 0) {
            cursor.close();
            return true;
        } else {
            cursor.close();
            return false;
        }


    }
```

5.3.8.7 public int getUserIdFromName(User user)

Η μέθοδος αυτή επιστρέφει από τη βάση την ταυτότητα του χρήστη, εφόσον ξέρουμε το όνομά του

```
public int getUserIdFromName(User user) {
        String query = "SELECT * FROM " + TABLE_USERS + " tu WHERE tu."
            + USER_NAME + " =?";
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(query, new String[]{user.
            getUserName()});
        int id = 0;
        if (cursor.moveToFirst()) {
            do {
                id = (cursor.getInt(cursor.getColumnIndex(USER_ID)));
            } while (cursor.moveToNext());
            user.setId(id);
        }
        return id;
    }
```

5.3.8.8 public byte[] getHmmFromUser(User user)

Η μέθοδος αυτή επιστρέφει από τη βάση το HMM που αντιστοιχεί στο χρήστη που περνιέται σαν όρισμα, σε μορφή πίνακα bytes

```
    public byte[] getHmmFromUser(User user){
        String query = "SELECT " + USER_HMM_BYTES + " FROM " +
            TABLE_USERS + " tu WHERE tu." + USER_ID
                + " =?";
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(
            user.getId())});


        if (cursor.moveToFirst()) {
            do {
                byte[] hmm = (cursor.getBlob(cursor.getColumnIndex(
                    USER_HMM_BYTES)));
                Log.d("HMM_FOUNT", hmm.toString());

                return hmm;
            } while (cursor.moveToNext());

        }

        return null ;

    }
```

5.3.8.9 public void exportDB()

H m'ejodos aut'h e'inai bohjhtik'h kai qrhsimopoie'itai gia na g'inei exagwg'h ths b'ashs gia thn kal'uterh epexergas'ia twn apotelesm'atwn ston H/U

```
    public byte[] getHmmFromUser(User user){
     public void exportDB() {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();
```

```
        FileChannel source = null;
        FileChannel destination = null;
        String currentDBPath = "/data/" + "com.example.christos.
            gestureappdemo" + "/databases/" + DATABASE_NAME;
        String backupDBPath = DATABASE_NAME;
        File currentDB = new File(data, currentDBPath);
        File backupDB = new File(sd, backupDBPath);
        try {
            source = new FileInputStream(currentDB).getChannel();
            destination = new FileOutputStream(backupDB).getChannel();
            destination.transferFrom(source, 0, source.size());
            source.close();
            destination.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Κεφάλαιο 6

ΜΕΘΟΔΟΛΟΓΙΑ ΕΛΕΓΧΟΥ

6.1 Περιγραφή

Για τον έλεγχο του συστήματος που δημιουργήθηκε ζητήθηκε από χρήστες η δοκιμή του, έτσι ώστε να εκτιμηθούν στη συνέχεια τα αποτελέσματα. Η διαδικασία συλλογής δεδομένων είχε ως εξής:

1. Ζητάμε από τον κάθε χρήστη να εισάγει μια χειρονομία ασφαλή και ευκολομνημόνευτη, όπως αντιλαμβάνεται ο ίδιος/ η ίδια τον όρο αυτό.

2. Στη συνέχεια ζητάμε να επαναλάβει τη χειρονομία άλλες 9 φορές έτσι ώστε να εκπαιδευτεί το HMM που θα αντιπροσωπεύει μοναδικά το χρήστη.

3. Ο χρήστης συμπληρώνει μια φόρμα εκτίμησης του φόρτου εργασίας που κατέβαλλε [8] και μετράει αντίστροφα από το 20 στο 0, έτσι ώστε να αποσπάσουμε την προσοχή του από την χειρονομία που μόλις εισήγαγε στο σύστημα.

4. Ο χρήστης εισάγει 2 χειρονομίες προς έλεγχο του σκορ τους με το δεδομένο HMM του.

5. Ο χρήστης μετά από 2-3 μέρες εισάγει άλλες 5 χειρονομίες στο σύστημα προς έλεγχο του σκορ τους με το δεδομένο HMM του.

Εφόσον το σύστημα έχει σχεδιαστεί σαν ένα σύστημα αυθεντικοποίησης χρηστών θα εξετασθεί η αποτελεσματικότητα του με βάση την επιθυμητή λειτουργία.

Όπως έχει αναφερθεί στην ενότητα με το θεωρητικό υπόβαθρο, ένα σύστημα αυθεντικοποίησης θα πρέπει να μηδενίζει τις λανθασμένες αυθεντικοποιήσεις (False Positive - FP) και να μηδενίζει ή έστω να περιορίζει τις λανθασμένες απορρίψεις (False Negative - FN). Το σύστημα που δημιουργήσαμε θα εξετασθεί ως προς αυτές τις παραμέτρους για διαφορετικά κατώφλια αυθεντικοποίησης. Επιπλέον για κάθε κατώφλι αυθεντικοποίησης θα βρεθούν και οι μετρικές που αναφέρθηκαν στο θεωρητικό κεφάλαιο, δηλαδή sensitivity (true positive rate), specificity (true negative rate), positive predictive value , negative predictive value , accuracy.

Εφόσον μας ενδιαφέρει η συνολική συμπεριφορά του συστήματος και όχι η συμπεριφορά ανά χειρονομία χρήστη, όλα τα δεδομένα θα παρουσιάζονται για όλο το σύστημα συνολικά.

6.2 Αναλυτική παρουσίαση

Για την εύρεση των λανθασμένων θετικών και των ορθώς αρνητικών αποτελεσμάτων υπολογίζουμε για κάθε HMM που έχουμε δημιουργήσει και υπάρχει στη βάση δεδομένων την πιθανότητα να έχουν προκύψει από αυτό όλες οι άλλες χειρονομίες οι οποίες δεν ανήκουν στον χρήστη που έχει δημιουργήσει το εκάστοτε HMM. Όσες από αυτές βγάλουν σκορ μεγαλύτερο από το εκάστοτε κατώφλι (threshold) θεωρούνται λανθασμένα θετικές απαντήσεις και όσες βγάζουν σκορ μικρότερο από το κατώφλι απόφασης θεωρούνται ορθά αρνητικές απαντήσεις.

Για την εύρεση των ορθώς θετικών αποτελεσμάτων και των λανθασμένων αρνητικών αποτελεσμάτων, εξετάζουμε τα σκορ των χειρονομιών των χρηστών με το δικό τους HMM. Όσα σκορ είναι μεγαλύτερα από το εκάστοτε κατώφλι είναι ορθώς θετικά αποτελέσματα και όσα είναι κάτω από το κατώφλι είναι λανθασμένα αρνητικά αποτελέσματα.

Ακολουθούν στον πίνακα 6.1 τα αποτελέσματα που παράγονται για τις διαφορετικές τιμές κατωφλίου. Όσο μεγαλύτερη τιμή κατωφλίου, τόσο πιο αυστηρό ως προς την αυθεντικοποίηση περιμένουμε να είναι το σύστημα μας.

threshold	FP	TN	TP	FN
-100	1	5704	4	123
-200	1	5704	20	107
-300	2	5703	52	75
-400	15	5690	75	52
-500	62	5643	87	40
-600	173	5532	97	30
-700	352	5353	100	27
-800	523	5182	101	26
-900	704	5001	104	23

Πίνακας 6.1: Αποτελέσματα για διαφορετικές τιμές κατωφλίου

Με βάση αυτά τα αποτελέσματα προκύπτουν οι τιμές των μετρικών που αναφέρθηκαν προηγουμένως 6.2

Στη συνέχεια ακολουθήσαμε την ίδια διαδικασία για την εύρεση των ορθών και λανθασμένων θετικών και αρνητικών απαντήσεων, αλλά αυτή τη φορά κανονικοποιήσαμε τα σκορ κάθε χειρονομίας διαιρώντας τα με το μήκος της χειρονομίας. Αυτή η τεχνική ακολουθήθηκε καθώς όσο μεγαλύτερη είναι μια χειρονομία, τόσο δυσκολότερο είναι να επιτευχθεί το επιθυμητό επίπεδο ακρίβειας από το χρήστη, επομένως μια διαίρεση με το μήκος της χειρονομίας θα μπορούσε να παράγει πιο δίκαια αποτελέσματα. Ακολουθούν στον πίνακα 6.3 τα αποτελέσματα που παράγονται για τις διαφορετικές τιμές κατωφλίου. Όσο μεγαλύτερη τιμή κατωφλίου, τόσο πιο αυστηρό ως προς την αυθε-

threshold	sensitivity(TPR)	specificity(TNR)	PPV	NPV	accuracy
-100	0.0007	0.9998	0.8000	0.9789	0.9787
-200	0.0035	0.9998	0.9524	0.9816	0.9815
-300	0.0090	0.9996	0.9630	0.9870	0.9868
-400	0.0130	0.9974	0.8333	0.9909	0.9885
-500	0.0152	0.9891	0.5839	0.9930	0.9825
-600	0.0172	0.9697	0.3593	0.9946	0.9652
-700	0.0183	0.9383	0.2212	0.9950	0.9350
-800	0.0191	0.9083	0.1619	0.9950	0.9059
-900	0.0204	0.8766	0.1287	0.9954	0.8753

Πίνακας 6.2: Μετρικές για διαφορετικές τιμές κατωφλίου

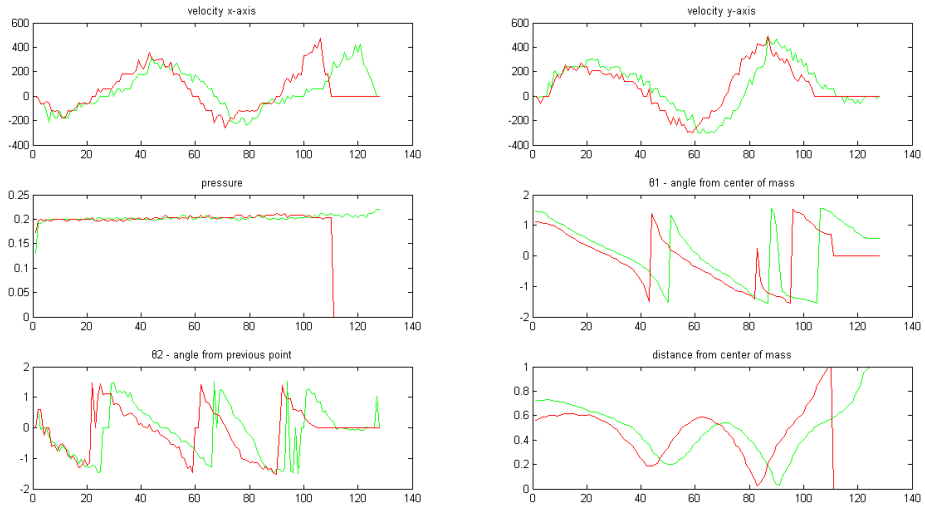
ντικοποίηση περιμένουμε να είναι το σύστημα μας.

threshold	FP	TN	TP	FN
-1	1	5704	5	122
-2	1	5704	25	102
-3	2	5703	44	83
-4	7	5698	66	61
-5	25	5680	79	48
-6	70	5635	89	38
-7	175	5530	94	33
-8	303	5402	98	29
-9	468	5237	98	29

Πίνακας 6.3: Αποτελέσματα για διαφορετικές τιμές κατωφλίου - κανονικοποιημένα σκορ

Με βάση αυτά τα αποτελέσματα προκύπτουν οι τιμές των μετρικών που αναφέρθηκαν

Σχήμα 6.1: Σύγκριση χαρακτηριστικών δύο χειρονομιών από διαφορετικού χρήστες



προηγουμένως 6.4

threshold	sensitivity(TPR)	specificity(TNR)	PPV	NPV	accuracy
-1	0.0008	0.9998	0.8333	0.9791	0.9789
-2	0.0044	0.9998	0.9615	0.9824	0.9823
-3	0.0077	0.9996	0.9565	0.9857	0.9854
-4	0.0115	0.9988	0.9041	0.9894	0.9883
-5	0.0137	0.9956	0.7596	0.9916	0.9875
-6	0.0155	0.9877	0.5597	0.9933	0.9815
-7	0.0167	0.9693	0.3494	0.9941	0.9643
-8	0.0178	0.9469	0.2444	0.9947	0.9431
-9	0.0184	0.9180	0.1731	0.9945	0.9148

Πίνακας 6.4: Μετρικές για διαφορετικές τιμές κατωφλίου-κανονικοποιημένα σκορ

Παρατηρώντας τα αποτελέσματα που προκύπτουν μπορούμε να βγάλουμε τα εξής συμπεράσματα για το σύστημα που δημιουργήθηκε. Αρχικά βλέπουμε ότι και για πολύ υψηλό κατώφλι έχουμε μια false positive χειρονομία. Βλέποντας πιο προσεκτικά την χειρονομία η οποία θεωρήθηκε ορθή ακόμα και με πολύ αυστηρό κριτήριο, θα δούμε ότι οι δύο διαφορετικοί χρήστες εισήγαγαν στο σύστημα δύο αρκετά όμοιες σε χαρακτηριστικά χειρονομίες. Στο σχήμα 6.1 βλέπουμε συγκεντρωμένα τα χαρακτηριστικά των 2 χειρονομιών συγκριτικά.

Εφόσον όπως βλέπουμε οι δύο χειρονομίες έχουν παρόμοια χαρακτηριστικά είναι λογικό να περνάει ακόμα και από το υψηλότερο κατώφλι το οποίο είναι το αυστηρότερο.

Συγκρίνοντας τα αποτελέσματα που προκύπτουν από τα κανονικοποιημένα σκορ, σε σχέση με τα αποτελέσματα που προέκυψαν χωρίς κανονικοποίηση με το μήκος της

κάθε χειρονομίας, παρατηρούμε ότι υπάρχει σαφώς μία σημαντική μείωση στα False positive αποτελέσματα και αντίστοιζα αύξηση στα True Negative αποτελέσματα για τα μεσαία κατώφλια, χρησιμοποιώντας κανονικοποίηση. Τα True Positive, False Negative αποτελέσματα παραμένουν σχεδόν ίδια και με τις δύο προσεγγίσεις.

Παρατηρώντας τις μετρικές του συστήματος βλέπουμε ότι και με τις δύο προσεγγίσεις το σύστημα παρουσιάζει αρκετά μικρή ευαισθησία η οποία αυξάνεται λίγο καθώς μειώνεται το κατώφλι απόφασης, γεγονός το οποίο σημαίνει ότι το σύστημα είναι πολύ επιλεκτικό στις χειρονομίες τις οποίες θεωρεί σαν θετικές, άρα ο χρήστης θα πρέπει να μπορεί να επαναλάβει σε μεγάλο βαθμό ομοιότητας τη χειρονομία του για να εισέλθει στο σύστημα. Επιπλέον βλέπουμε ότι το σύστημα παρουσιάζει σχεδόν ιδανική ειδικότητα (specificity) για υψηλές τιμές κατωφλίου που πλησιάζει τη μονάδα, γεγονός που σημαίνει ότι το σύστημα μπορεί να αναγνωρίσει με μεγάλη ακρίβεια ποιες χειρονομίες δεν έχουν παραχθεί από το χρήστη που δημιούργησε το μοντέλο. Οι μετρικές PPV, NPV δείχνουν ότι για αυστηρά προς μέτρια κατώφλια απόφασης (-2,-3) μπορούμε να έχουμε μεγαλύτερο ποσοστό θετικών αποτελεσμάτων που έχουν καταχωρηθεί σωστά, ενώ το ποσοστό των αρνητικών αποτελεσμάτων που ορθώς δεν καταχωρήθηκαν στην υπό εξέταση κλάση αυξάνεται όσο το κατώφλι μειώνεται.

Γενικά το σύστημα πετυχαίνει και με τις δύο προσεγγίσεις καλύτερη ακρίβεια με κατώφλι απόφασης -400 και -4 αντίστοιχα. Σε εκείνο το επίπεδο κατωφλίου παρατηρούμε ότι μειώνονται τα False Negative αποτελέσματα και ταυτόχρονα αυξάνονται τα True Positive με μικρότερο ρυθμό από την αύξηση των False Positive αποτελεσμάτων επομένως παίρνουμε την καλύτερη δυνατή τιμή ακρίβειας.

Κεφάλαιο 7

ΕΠΙΛΟΓΟΣ

7.1 Σύνοψη και Συμπεράσματα

Στα πλαίσια αυτής της διπλωματικής δημιουργήθηκε μία εφαρμογή για Android smartphones η οποία συλλέγει χειρονομίες χρηστών και με βάση αυτές δημιουργεί ένα HMM για κάθε χρήστη. Στη συνέχεια υπολογίζει το σκορ νέων χειρονομιών κάθε χρήστη δεδομένου του μοντέλου του. Όλα τα δεδομένα συλλέγονται και αποθηκεύονται σε μία βάση δεδομένων.

Τα συμπεράσματα που προκύπτουν από την διπλωματική είναι ότι το σύστημα που δημιουργήθηκε και βασίζεται πάνω στα Κρυφά Μαρκοβιανά Μοντέλα (HMM) μπορεί να χρησιμοποιηθεί επιτυχώς για την αναγνώριση χειρονομιών και την κατηγοριοποίηση των χρηστών που τις δημιούργησαν.

Επιπλέον, εφόσον το σύστημα αναπτύχθηκε εξ΄ ολοκλήρου πάνω σε κινητό τηλέφωνο δεν απαιτεί πολλούς υπολογιστικούς πόρους και θα μπορούσε να χρησιμοποιηθεί για την αυθεντικοποίηση χρηστών για διάφορες εφαρμογές.

7.2 Μελλοντικές επεκτάσεις

Πιθανές μελλοντικές επεκτάσεις της εφαρμογής που αναπτύχθηκε είναι η προσαρμογή της για αναγνώριση χειρονομιών οι οποίες θα λαμβάνονται με άλλα μέσα και όχι μέσω της οθόνης αφής, καθώς επίσης και η εξέταση εναλλακτικών χαρακτηριστικών για την μοντελοποίηση των χειρονομιών.

Οι χειρονομίες θα μπορούσαν να λαμβάνονται μέσω της κάμερας του τηλεφώνου. Μια πολύ ενδιαφέρουσα εφαρμογή θα ήταν η αναγνώριση νοηματικής γλώσσας με βάση τις χειρονομίες που θα καταγράφονται από την κάμερα του τηλεφώνου και η απευθείας μετάφραση στον χρήστη.

Βιβλιογραφία

[1] Edson J. R. Justino and Abdenaim El Yacoubi and Flávio Bortolozzi and Robert Sabourin, "An off-line signature verification system using hidden markov model and cross-validation," in *13th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2000), 17-20 October 2000, Gramado (RS), Brazil*, pp. 105–112, 2000.

[2] J. Fierrez, J. Ortega-Garcia, D. Ramos, and J. Gonzalez-Rodriguez, "Hmm-based on-line signature verification: Feature extraction and signature modeling," *Pattern Recogn. Lett.*, vol. 28, pp. 2325–2334, Dec. 2007.

[3] Wikipedia, the free encyclopedia, "Polynomial regression," 2015. [Online; accessed 3-September-2015].

[4] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 881–892, July 2002.

[5] J.-M. Francois, "Jahmm, an implementation of hidden markov models in java," 2009.

[6] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[7] V. Faber, "Clustering and the continuous k-means algorithm," *Los Alamos Science*, vol. 22, pp. 138–144, 1994.

[8] S. Hart, "Nasa task load index,"

[9] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *PROCEEDINGS OF THE IEEE*, pp. 257–286, 1989.

[10] H. Shu, "On-line handwriting recognition using hidden markov models."

[11] F. sheng Chen, C. ming Fu, and C. lin Huang, "y huang, c.: Hand gesture recognition using a real-time tracking method and hidden markov models," *Image and Video Computing*, pp. 745–758, 2003.

[12] M. Sherman, G. Clark, Y. Yang, S. Sugrim, A. Modig, J. Lindqvist, A. Oulasvirta, and T. Roos, "User-generated free-form gestures for authentication: Security and memorability," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, (New York, NY, USA), pp. 176–189, ACM, 2014.