Πανεπιστήμιο Πελοποννήσου
Τμήμα Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή Matlab για την ανίχνευση στατιστικών διαφορών χαρακτηριστικών εικονοστοιχείων για την ανίχνευση προσώπων και πεζών»

Ανδρεοπούλου Ειρήνη – Α.Μ. 1863

Ζόγκζας Γεώργιος – Α.Μ. 1913

ΕΠΙΒΛΕΠΩΝ: Μιχάλης Παρασκευάς,
Αναπληρωτής Καθηγητής

ΠΑΤΡΑ 2020

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα, 2020.

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Μιχάλης Παρασκευάς
2. Τζήμας Ιωάννης
3. Χριστοδούλου Σωτήρης

Ευχαριστίες

Η παρούσα πτυχιακή εργασία είναι το αποτέλεσμα μιας σειράς αλληλεπιδράσεων με ένα πλήθος ατόμων, καθένα από τα οποία έπαιξαν ένα σημαντικό ρόλο στην εξέλιξη και την ολοκλήρωσή της. Αξίζει, λοιπόν, αυτή η σελίδα να αφιερωθεί σε όλους αυτούς, εκφράζοντας τους ένα μεγάλο ευχαριστώ.

Αρχικά, θέλουμε να ευχαριστήσουμε τον υπεύθυνο καθηγητή μας κ. Μιχάλη Παρασκευά που μας εμπιστεύτηκε και μας έδωσε τη δυνατότητα να συγγράψουμε την παρούσα πτυχιακή εργασία. Επίσης τις ευχαριστίες μας θα θέλαμε να τις εκφράσουμε προς όλους τους καθηγητές του τμήματος Μηχανικών Πληροφορικής του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Δυτικής Ελλάδας για τις πολύτιμες γνώσεις που μας προσέφεραν όλα αυτά τα χρόνια.

Τέλος, ένα μεγάλο ευχαριστώ στα αγαπημένα μας πρόσωπα, τους γονείς μας, τα αδέρφια μας και τους κολλητούς μας φίλους που με την πίστη τους σε εμάς, την υπομονή τους ορισμένες φορές και την αμέριστη συμπαράσταση και υποστήριξη τους σε κάθε φάση της ζωής μας.

Περιεχόμενα

Περίληψη

Με την πάροδο του χρόνου η κοινωνία μας αναπτύσσεται και εξελίσσεται ταυτόχρονα. Μαζί με την κοινωνία όμως αναπτύσσονται οι ανάγκες και οι απαιτήσεις της. Η τεχνολογία τα τελευταία χρόνια έχει υποστεί ραγδαία εξέλιξη με σκοπό την εξυπηρέτηση αυτών των αναγκών. Η τεχνολογία και κυρίως η ανάπτυξη των υπολογιστών έχει εισχωρήσει για τα καλά στην καθημερινότητα μας.

Στην παρούσα πτυχιακή εργασία θα ασχοληθούμε με ένα από τα σημαντικότερα θέματα της ανάπτυξης των υπολογιστών, το οποίο δεν είναι άλλο παρά από την μηχανική μάθηση. Πιο συγκεκριμένα θα ασχοληθούμε με την ανάπτυξη αλγορίθμων με στόχο την ανίχνευση και την αναγνώριση ανθρώπων (πεζών) σε φωτογραφίες από κάμερες κυκλοφορίας. Το προγραμματιστικό περιβάλλον που θα χρησιμοποιήσουμε είναι το Matlab, Στο πρόγραμμα μας χρησιμοποιούμε τρεις αλγόριθμους και τρεις ταξινομητές. Οι αλγόριθμοι που χρησιμοποιούμε είναι ο HOG (Histogram of Oriented Gradients), ο PCA (Principal Component Analysis) και ο LBP (Local Binary Patterns). Οι ταξινομητές που χρησιμοποιούμε είναι ο Random Forest, ο SVM (Support Vector Machine) και ο Gaussian Process. Επίσης στην συνέχεια της πτυχιακής μας αναλύουμε τον τρόπο λειτουργίας των αλγορίθμων και των ταξινομητών που χρησιμοποιούμε για την υλοποίηση του προγράμματος μας. Εκτός από όλα τα παραπάνω θα αναφερθούμε και σε άλλα θέματα / βασικές ορολογίες που σχετίζονται με το θέμα της πτυχιακής μας όπως είναι η εικόνα, οι φωτογραφικές μηχανές, ο θόρυβος και η ανίχνευση ανθρώπων. Τέλος βλέπουμε διεξοδικά τον κώδικα μας βήμα βήμα το ως προς τον τρόπο λειτουργίας του, καθώς επίσης και τα αποτελέσματα που παίρνουμε απ’ αυτόν, όπως και που αλλού θα μπορούσε να χρησιμοποιηθεί ο ίδιος ή παρόμοιος κώδικας.

Abstract

Over time, our society develops and evolves at the same time. Together with society, however, its needs and requirements are developing. Technology in recent years has undergone rapid development to serve these needs. Technology and especially the development of computers has entered our daily lives for good.

In this dissertation we will deal with one of the most important issues in the development of computers, which is nothing but mechanical learning. More specifically, we will deal with the development of algorithms aimed at detecting and identifying people (pedestrians) in photos from traffic cameras. The programming environment we will use is Matlab. In our program we use three algorithms and three classifiers. The algorithms we use are HOG (Histogram of Oriented Gradients), PCA (Principal Component Analysis) and LBP (Local Binary Patterns). The classifiers we use are Random Forest, SVM (Support Vector Machine) and Gaussian Process. Also, in the continuation of our dissertation, we analyze the mode of operation of the algorithms and classifiers that we use for the implementation of our program. In addition to all of the above, we will refer to other topics / basic terminology related to the subject of our degree, such as image, cameras, noise and human detection. Finally, we look at our code in detail step by step in terms of how it works, as well as the results we get from it, as well as where else it could be used itself or a similar code.

Κεφάλαιο 1

1.1. Εισαγωγή

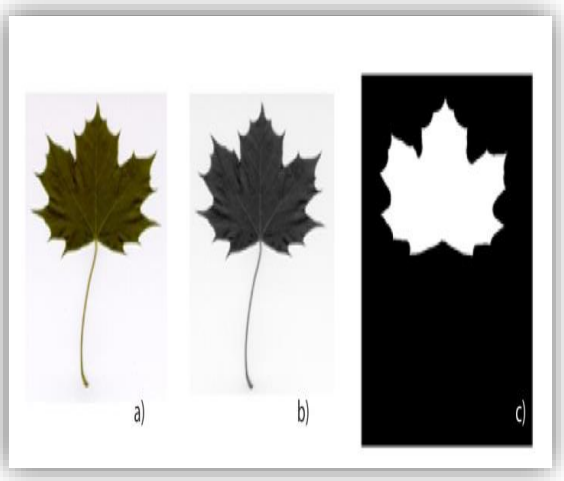
Σε αυτό το κεφάλαιο θα ασχοληθούμε σχετικά με τις ορολογίες των βασικών κομματιών που χρησιμοποιήσαμε για την υλοποίηση της παρακάτω πτυχιακής, των οποίων ο συνδυασμός όλων αυτών μας έδωσε το παρακάτω αποτέλεσμα. Μια από τις πιο βασικές έννοιες που θα αναλύσουμε παρακάτω είναι αυτή της Μηχανικής Μάθησης, όπως επίσης και οι εικόνες, η ανίχνευση ανθρώπων, οι φωτογραφικές μηχανές, ο θόρυβος, όπως και μια μικρή σύσταση του προγράμματος που χρησιμοποιήσαμε (Matlab) για την διεκπεραίωση του κώδικα μας.

1.2. Εικόνα

Ως εικόνα ορίζουμε την αποτύπωση ενός ή περισσότερων αντικειμένων καθώς και το περιβάλλοντα χώρου. Αυτή η αποτύπωση μπορεί να γίνει είτε σε αναλογική είτε σε ψηφιακή μορφή. Οι εικόνες μπορούν να αποτυπωθούν σε χαρτί, στην μνήμη ή στην οθόνη ενός υπολογιστή καθώς και σε άλλα μέσα. Μια εικόνα μπορεί να είναι μια ανάμνηση ή ένα προϊόν φαντασίας.

Μια εικόνα όταν βρίσκεται σε ψηφιακή μορφή έχει τη δυνατότητα της επεξεργασίας, δηλαδή την αλλαγή των αρχικών χαρακτηριστικών της. Συνήθως η επεξεργασία μιας εικόνας γίνεται για την βελτίωση της πληροφορίας της ανάλογα κρίση αυτού που την επεξεργάζεται.

Οι ψηφιακές εικόνες χωρίζονται σε τρεις κατηγορίες: τις **Binary**, τις **Grayscale** και τις **True Color** (**RGB**). Οι Binary εικόνες αποτελούνται από δύο τιμές το 0 και το 1. Αυτές οι τιμές στην αποτύπωση τους η μια είναι άσπρο (1) και η άλλη μαύρο (0). Οι Grayscale εικόνες αποτελούνται από 256 αποχρώσεις του γκρι, όπου το μαύρο έχει την τιμή 0 και το άσπρο την τιμή 255. Τέλος οι RGB εικόνες χωρίζονται σε τρία επίπεδα όπως φαίνεται και στο όνομα τους R(Red) G(Green) B(Blue), κάθε ένα επίπεδο μπορεί να πάρει τιμή από το 0 έως το 255 όπου αυτό έχει σαν αποτέλεσμα να έχουμε 255^3= 16.777.216 διαφορετικά πιθανά χρώματα.



Εικόνα 1.1 a) RCB b) Grayscale c) Binary

Μια μεγάλη κατηγορία που ανήκει μέσα στις εικόνες είναι οι φωτογραφίες. Όταν λέμε φωτογραφία αναφερόμαστε στην τέχνη και την επιστήμη της δημιουργίας οπτικών εικόνων μέσω της αποτύπωσης του φωτός με την χρήση κατάλληλων συσκευών (φωτογραφικές μηχανές), τις οποίες συσκευές θα τις δούμε σε μετέπειτα

ενότητα. Η φωτογραφία εκτός από την τεχνολογική της διάσταση αναγνωρίζεται και ως ένα από τα ευρύτερα διαδεδομένα μέσα επικοινωνίας, όπως επίσης και ως μια μορφής τέχνης. Από τις αρχές του 18ου αιώνα είχαν ξεκινήσει να γίνονται προσπάθειες για την αποτύπωση της πρώτης φωτογραφίας. Στην αρχή στις πρώτες προσπάθειες δεν κατάφερναν να την αποτυπώσουν σε χαρτί, μετέπειτα κατάφεραν να αποτυπώνουν το αρνητικό της φωτογραφίας και ύστερα από μερικά χρόνια κατάφεραν να τυπώσουν και το θετικό της. Οι πρώτες φωτογραφίες που βγήκαν ήταν ασπρόμαυρες, και ύστερα σχεδόν έναν αιώνα κατάφεραν να τυπώσουν τις πρώτες έγχρωμες φωτογραφίες.

Με το πέρας του χρόνου είχαμε και την εξέλιξη της φωτογραφίας από αυτές με φιλμ σε ψηφιακές φωτογραφίες. Η ψηφιακή φωτογραφία αποτελεί ίσως την τελευταία σημαντική εξέλιξη σε ό,τι αφορά την τεχνική της φωτογραφίας. Στην ψηφιακή φωτογραφία, αντί για το κοινό φιλμ, χρησιμοποιούνται φωτοευαίσθητοι αισθητήρες. Ορισμένες συνηθισμένες μορφές-τύποι αποθήκευσης σε ψηφιακά μέσα είναι οι: jpeg, tiff, bmp, gif, png. Τα μέσα αναπαραγωγής της εικόνας είναι οι ίδιες οι φωτογραφικές μηχανές, οι οθόνες των ηλεκτρονικών υπολογιστών αλλά και μυριάδες μέσα ψηφιακής απεικόνισης. Για κάθε ένα από αυτά, χρειάζεται η προσαρμογή της μορφής καταγραφής στις απαιτήσεις του συστήματος. Επίσης μπορούμε να πούμε ότι η φωτογραφίες χωρίζονται σε είδη (κατηγορίες) τα οποία είναι η Φωτοειδησεογραφία η οποία αφορά την επικαιρότητα, η Διαφημιστική φωτογραφία ή φωτογραφία στούντιο η οποία αφορά αντικείμενα, μόδα, πορτραίτα, η Αρχιτεκτονική φωτογραφία ή φωτογραφία Εσωτερικών Χώρων η οποία σχετίζεται με κτήρια και εσωτερικούς χώρους, η φωτογραφία Τέχνης η οποία έχει σχέση με την φωτογραφία ως αυτόνομη και ανεξάρτητη τέχνη και τέλος η Αστροφωτογραφία η οποία έχει να κάνει με την απεικόνιση αντικειμένων του νυχτερινού ουρανού (άστρα, νεφελώματα, γαλαξίες). Τέλος τα τελευταία χρόνια μετά την αποτύπωση της φωτογραφίας στο φιλμ πρέπει να ακολουθήσει η εμφάνιση του φιλμ, δηλαδή η επεξεργασία του για τη σταθεροποίηση της εικόνας, ώστε να μπορεί να εκτεθεί στο φως χωρίς να καταστραφεί. Στη συνέχεια, γίνεται η εκτύπωση κάθε καρέ σε φωτογραφικό χαρτί, δηλαδή χαρτί με κατάλληλες φωτοευαίσθητες επιστρώσεις. Αυτή γίνεται με την προβολή της εικόνας του αρνητικού πάνω στο εν λόγω χαρτί. Κατά τη διάρκεια αυτών των εργασιών είναι δυνατόν να γίνουν μια σειρά από παρεμβάσεις που μεταβάλλουν το τελικό αποτέλεσμα, δίνοντας ξεχωριστό χαρακτήρα.

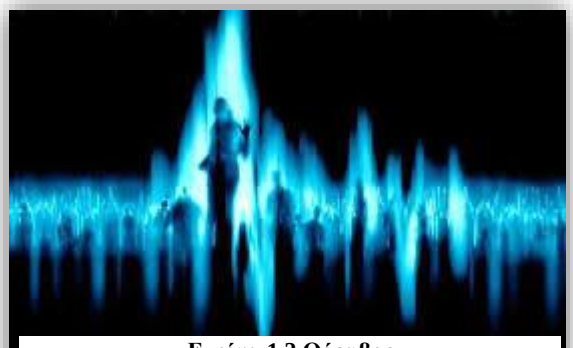
Για τις ψηφιακές φωτογραφίες, είναι δυνατή η επεξεργασία με ψηφιακά μέσα. Αυτό μπορεί να γίνει είτε μέσα στην φωτογραφική μηχανή, είτε σε ηλεκτρονικό υπολογιστή. Για τον σκοπό αυτό, έχει αναπτυχθεί μία μεγάλη σειρά από προγράμματα.

1.3. Θόρυβος

Ένα από τα βασικότερα προβλήματα την ψηφιακή επεξεργασία σήματος αλλά και στις εικόνες είναι ο θόρυβος. Ως θόρυβο μπορούμε να θεωρήσουμε μια μεταβολή στην τιμή του σήματος μας από την πραγματική του τιμή. Στις περισσότερες εικόνες

αν όχι σε όλες από την αρχή που ξεκίνησαν να αποτυπώνονται υπάρχει θόρυβος. Ο θόρυβος υπάρχει παντού στην καθημερινή μας ζωή, στις εικόνες τον συναντάμε σε διάφορες μορφές.

Θόρυβο σε μια εικόνα μπορούμε να χαρακτηρίσουμε το οτιδήποτε μπορεί να μας αλλοιώσει την πληροφορία μας, π.χ σε μια εικόνα μπορεί να χαρακτηρίσουμε σαν θόρυβο την ομίχλη, τη σκόνη, τη βροχή, το κενό, αλλά σε μια άλλη εικόνα μπορεί κάποιο από αυτά να είναι η πληροφορία μας. Επίσης όταν θέλουμε να ψηφιοποιήσουμε μια εικόνα ο μετατροπέας εισάγει θόρυβο στην εικόνα μας.



Εικόνα 1.2 Θόρυβος

Τέλος όταν έχουμε μια εικόνα όπου έχει σφάλματα μπορούμε να εισάγουμε τεχνικό θόρυβο με διάφορα φίλτρα θορύβου έτσι ώστε να γίνει εξομάλυνση της περιοχής αυτής που θέλουμε να διορθώσουμε, έτσι ώστε η εικόνα να είναι πιο «καλή» από ότι ήταν αρχικά. Τα φίλτρα που χρησιμοποιούμε σε μια εικόνα χωρίζονται σε γραμμικά και μη γραμμικά. Επιπλέον ένα σημαντικό χαρακτηριστικό που υπάρχει στις εικόνες είναι συνδεσιμότητα, δηλαδή το να μπορούμε να δούμε ποια pixels «γειτονεύουν» με ποια. Όταν γίνεται η εφαρμογή ενός φίλτρου σε μια εικόνα βασίζεται στην έννοια της «γειτονίας», δηλαδή η τιμή για κάθε pixel (target) στην εικόνα παίρνει μια νέα τιμή η οποία εξαρτάται από την τιμή των pixels που βρίσκονται στη γειτονιά γύρω από το target pixel.

1.4. Φωτογραφικές Μηχανές

Ως φωτογραφική μηχανή μπορούμε να χαρακτηρίσουμε τη συσκευή που χρησιμοποιούμε για να βγάλουμε φωτογραφίες. Οι πιο διαδεδομένες μηχανές (ερασιτεχνικές ή επαγγελματικές) χωρίζονται σε δυο βασικές κατηγορίες τις συμπαγείς (compact) και στις μονοοπτικές ρεφλέξ (SLR). Οι οποίες ανάλογα με την τεχνολογία χωρίζονται σε **φωτογραφικές με φιλμ** και σε **ψηφιακές φωτογραφικές**.

Εικόνα 1.3 Ψηφιακή φωτογραφική μηχανή και φωτογραφική μηχανή με φιλμ

Οι φωτογραφικές με φιλμ όπως λέει και το όνομα τους είναι αυτές που χρησιμοποιούν φιλμ, το οποίο φιλμ περνάει από τη διαδικασία εμφάνισης σε σκοτεινό θάλαμο ή σε ειδικά φωτογραφικά εργαστήρια όπου γίνονται οι απαραίτητες ενέργειες για την εμφάνιση του φιλμ.

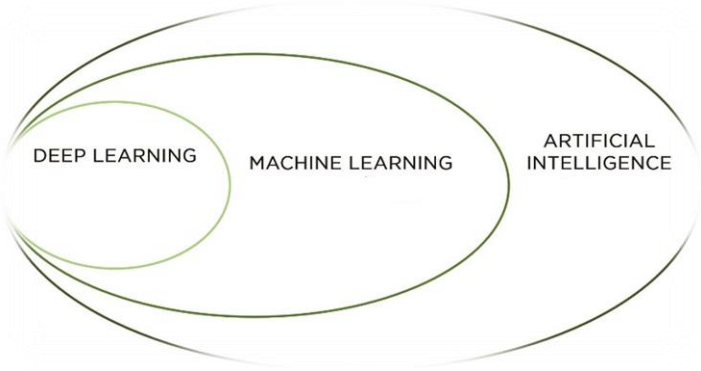
Οι ψηφιακές φωτογραφικές είναι αυτές που χρησιμοποιούν αισθητήρες εικονοστοιχείων για την καταγραφή και κάρτα μνήμης για την αποθήκευση, όπου στη συνέχεια μπορούν να τυπωθούν σε χαρτί στα φωτογραφικά εργαστήρια ή από ιδιωτικούς εκτυπωτές, ή επίσης μπορούν να μείνουν σε ηλεκτρονική μορφή.

1.5. Μηχανική Μάθηση

Η Μηχανική Μάθηση είναι μια υποκατηγορία του κλάδου της επιστήμης των υπολογιστικών συστημάτων. Στα τέλη της δεκαετίας του ’50 ο Arthur Samuel όρισε ότι η Μηχανική Μάθηση είναι η τέχνη των υπολογιστών να έχουν την δυνατότητα να αυτοδιδάσκονται. Η τέχνη της Μηχανικής Μάθησης όταν πρωτοξεκίνησε είχε εμπλακεί σε πολύ μεγάλο βαθμό με τη Τεχνητή Νοημοσύνη. Το ’80 υπήρξε ο διαχωρισμός μεταξύ της Μηχανικής Μάθησης και της Τεχνητής Νοημοσύνης. Η επιστήμη της Μηχανικής Μάθησης έπειτα από το διαχωρισμό από την Τεχνητή Νοημοσύνη άρχισε να ανθίζει κατά τη δεκαετία του ’90, όπου μέχρι και σήμερα βρίσκεται στην ακμή της.

Εικόνα 4 Μηχανική Μάθηση

Παρόλο τον διαχωρισμό τους θεωρούμε την **Τεχνική Νοημοσύνη** (Artificial Intelligence) το ευρύ κομμάτι αυτού του κλάδου ο οποίος διαχωρίζεται σε επιμέρους στάδια εκβάθυνσης το αμέσως επόμενο στάδιο είναι η **Μηχανική Μάθηση** (Machine Learning)και το αμέσως επόμενο είναι η **Βαθιά Μάθηση** (Deep Learning). Παρακάτω θα δούμε τι περιέχει το κάθε στάδιο.



Εικόνα 1.5 Στάδια Τεχνικής Νοημοσύνης

Τεχνητή Νοημοσύνη

Ως Τεχνητή Νοημοσύνη χαρακτηρίζεται η μελέτη των “ευφυών πρακτόρων”, με λίγα λόγια θα μπορούσαμε να πούμε ότι είναι η τέχνη κάθε συσκευή να αντιλαμβάνεται το περιβάλλον της και να αναλαμβάνει ενέργειες που μεγιστοποιούν την πιθανότητα επιτυχίας της επίτευξης των στόχων της. Ένας ακόμη πιο επίσημος ορισμός θα μπορούσαμε να πούμε ότι είναι η ικανότητα ενός συστήματος να ερμηνεύει σωστά τα

εξωτερικά δεδομένα, να μαθαίνει από τέτοια δεδομένα και να χρησιμοποιεί αυτά τα μαθήματα για την επίτευξη συγκεκριμένων στόχων και εργασιών μέσω ευέλικτης προσαρμογής. Η Τεχνητή Νοημοσύνη τις περισσότερες φορές περιστρέφεται γύρω από την χρήση αλγορίθμων, όπου ένας αλγόριθμος είναι ένα σύνολο ξεκάθαρων οδηγιών που μπορεί να εκτελέσει ένας μηχανικός υπολογιστής, ένας σύνθετος αλγόριθμος δημιουργείται συχνά πάνω από άλλους απλούστερους αλγόριθμους. Βέβαια η επιστήμη της Τεχνητής Νοημοσύνης δεν ήταν πάντα έτσι, στα τέλη της δεκαετίας το ’80 και της δεκαετίας του ’90 η Τεχνητή Νοημοσύνη είχε αναπτύξει μεθόδους για την αντιμετώπιση αβέβαιων ή ελλιπών πληροφοριών χρησιμοποιώντας έννοιες από τις πιθανότητες και τα οικονομικά. Αυτοί όμως οι αλγόριθμοι αποδείχθηκαν ανεπαρκείς για την επίλυση συλλογισμού διότι έγιναν εκθετικά πιο αργοί καθώς τα προβλήματα μεγάλωναν, όπου είχε σαν αποτέλεσμα έπειτα από χρόνια να λυθεί αυτό το πρόβλημα. Στην Τεχνητή Νοημοσύνη παρέμειναν τα εμπειρικά συστήματα, τα οποία μιμούνταν τον τρόπο σκέψης των ερευνητών για τη λήψη αποφάσεων και δεν αποκτούσαν δικό τους τρόπο σκέψης.

Μηχανική Μάθηση

Ως Μηχανική Μάθηση θεωρούμε ένα υποπεδίο της Τεχνητής Νοημοσύνης, όπου για πρώτη φορά το 1959 ο Άρθουρ Σάμουελ έδωσε τον ορισμό της ως « Το πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί». Η μηχανική μάθηση διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα. Παρόλα αυτά στη συνέχεια ο Tom Mitchell έδωσε έναν πιο επίσημο ορισμό για την Μηχανική Μάθηση ο οποίος πλέον χρησιμοποιείται ευρέως, ο ορισμός έλεγε ότι «Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία Ε ως προς μια κλάση εργασιών Τ και ένα μέτρο επίδοσης Ρ, αν η επίδοση του σε εργασίες της κλάσης Τ, όπως αποτιμάται από το μέτρο Ρ, βελτιώνεται με την εμπειρία Ε».

Η Μηχανική Μάθηση συνήθως ταξινομεί τις εργασίες της σε τρεις μεγάλες κατηγορίες, ανάλογα με τη φύση του εκπαιδευτικού “σήματος” που διαθέσιμα σε ένα σύστημα εκμάθησης. Από αυτές τις τρεις κατηγορίες η πρώτη είναι η **Επιτηρούμενη Μάθηση** όπου το υπολογιστικό πρόγραμμα δέχεται τις παραδειγματικές εισόδους καθώς και τα επιθυμητά αποτελέσματα από έναν “δάσκαλο”, και ο στόχος είναι να μάθει έναν γενικό κανόνα προκειμένου να αντιστοιχίσει τις εισόδους με τα αποτελέσματα. Η δεύτερη είναι η **Μη Επιτηρούμενη Μάθηση** όπου χωρίς να παρέχεται κάποια εμπειρία στον αλγόριθμο μάθησης, πρέπει να βρει την δομή των δεδομένων εισόδου, επίσης μπορεί να είναι αυτοσκοπός ή μέσος για ένα τέλος. Και η τρίτη είναι η **Ενισχυτική Μάθηση**, όπου ένα πρόγραμμα αλληλεπιδρά με ένα δυναμικό περιβάλλον στον οποίο πρέπει να επιτευχθεί ένας συγκεκριμένος στόχος, χωρίς κάποιος “δάσκαλος” να του λέει ρητά αν έχει φτάσει κοντά στο στόχο του.

Επίσης η Μηχανική Μάθηση συνδέεται και με την βελτιστοποίηση, πολλά προβλήματα μάθησης διατυπώνονται ως η ελαχιστοποίηση της συνάρτησης απώλειας από ένα σύνολο δεδομένων εκπαίδευσης. Η συνάρτηση απώλειας εκφράζει τη διαφορά μεταξύ των προβλέψεων του εκπαιδευμένου μοντέλου και των πραγματικών καταστάσεων του προβλήματος. Η διαφορά των δύο τομέων απορρέει από τον στόχο της γενίκευσης: ενώ οι αλγόριθμοι βελτιστοποίησης μπορούν να ελαχιστοποιήσουν την απώλεια ενός συνόλου εκπαίδευσης, η μηχανική μάθηση εστιάζει στην ελαχιστοποίηση της απώλειας σε άγνωστες καταστάσεις.

Βαθιά Μάθηση

Η Βαθιά Μάθηση είναι μέρος μιας ευρύτερης οικογένειας μεθόδων η οποία είναι μια υποκατηγορία της Μηχανικής Μάθησης, η οποία είναι κατάλληλη για αλγορίθμους αυτό-εκπαίδευσης και εξαγωγής μελλοντικών αποτελεσμάτων. Η Βαθιά Μάθηση μπορεί να επιβλέπεται, να ημι-επιβλέπεται ή να είναι χωρίς επίβλεψη. Το επίθετο “deep” στη Βαθιά Μάθηση προέρχεται από τη χρήση πολλαπλών επιπέδων στο δίκτυο. Η βαθιά μάθηση είναι μια σύγχρονη παραλλαγή που αφορά έναν απεριόριστο αριθμό επιπέδων οριοθετημένου μεγέθους, το οποίο επιτρέπει την πρακτική εφαρμογή και τη βελτιστοποιημένη εφαρμογή, διατηρώντας παράλληλα τη θεωρητική καθολικότητα υπό ήπιες συνθήκες. Στη βαθιά μάθηση, τα στρώματα επιτρέπεται επίσης να είναι ετερογενή και να αποκλίνουν ευρέως από βιολογικά ενημερωμένα μοντέλα σύνδεσης , για χάρη της αποτελεσματικότητας, της δυνατότητας εκπαίδευσης και της κατανόησης, από όπου το «δομημένο» μέρος. Τα περισσότερα σύγχρονα μοντέλα βαθιάς μάθησης βασίζονται σε τεχνητά νευρωνικά δίκτυα. Στη βαθιά μάθηση, κάθε επίπεδο μαθαίνει να μετατρέπει τα δεδομένα εισόδου του σε μια ελαφρώς πιο αφηρημένη και σύνθετη αναπαράσταση. Σε μια εφαρμογή αναγνώρισης εικόνας, η ακατέργαστη είσοδος μπορεί να είναι μια μήτρα εικονοστοιχείων. Το πρώτο αντιπροσωπευτικό στρώμα μπορεί να αφαιρέσει τα εικονοστοιχεία και να κωδικοποιεί τις άκρες. Το δεύτερο στρώμα μπορεί να συνθέτει και να κωδικοποιεί διευθετήσεις άκρων. το τρίτο στρώμα μπορεί να κωδικοποιεί μια μύτη και τα μάτια. και το τέταρτο επίπεδο μπορεί να αναγνωρίσει ότι η εικόνα περιέχει ένα πρόσωπο. Είναι σημαντικό ότι μια διαδικασία βαθιάς μάθησης μπορεί να μάθει ποια χαρακτηριστικά να τοποθετήσει βέλτιστα σε ποιο επίπεδο από μόνη της.

Παρόλα αυτά θα πρέπει να διερωτηθούμε, γιατί να χρησιμοποιούμε μηχανική μάθηση; Ας σκεφτούμε ότι θα γράφαμε ένα φίλτρο spam χρησιμοποιώντας παραδοσιακές τεχνικές προγραμματισμού

1. Πρώτα θα δούμε πώς μοιάζει συνήθως το spam. Μπορεί να παρατηρήσετε ότι ορισμένες λέξεις ή φράσεις τείνουν να εμφανίζονται πολύ στο θέμα. Ίσως θα παρατηρήσετε επίσης μερικά άλλα μοτίβα στο όνομα του αποστολέα, το σώμα του μηνύματος ηλεκτρονικού ταχυδρομείου και ούτω καθεξής.
2. Θα γράφατε έναν αλγόριθμο ανίχνευσης για καθένα από τα μοτίβα που

παρατηρήσατε και το πρόγραμμά σας θα επισημαίνει τα μηνύματα ηλεκτρονικού ταχυδρομείου ως ανεπιθύμητα αν εντοπιστούν ορισμένα από αυτά τα μοτίβα.

3. Θα δοκιμάσετε το πρόγραμμά σας και θα επαναλάβετε τα βήματα 1 και 2 έως ότου είναι αρκετά καλό. Η παραδοσιακή προσέγγιση Δεδομένου ότι το πρόβλημα δεν είναι ασήμαντο, το πρόγραμμά σας πιθανότατα θα γίνει μια μεγάλη λίστα περίπλοκων κανόνων - πολύ δύσκολο να διατηρηθεί. Αντίθετα, ένα φίλτρο ανεπιθύμητης αλληλογραφίας που βασίζεται σε τεχνικές μηχανικής εκμάθησης μαθαίνει αυτόματα ποιες λέξεις και φράσεις είναι καλές προβλέψεις του ανεπιθύμητου περιεχομένου εντοπίζοντας ασυνήθιστα συχνά μοτίβα λέξεων στα παραδείγματα ανεπιθύμητων μηνυμάτων σε σύγκριση με τα παραδείγματα ζαμπόν. Το πρόγραμμα είναι πολύ μικρότερο, πιο εύκολο στη συντήρηση και πιθανότατα πιο ακριβές.

Επίσης θα δούμε μερικά συγκεκριμένα παραδείγματα εργασιών μηχανικής μάθησης, μαζί με τις τεχνικές που μπορούν να αντιμετωπίσουν:

1. Ανίχνευση όγκων σε σαρώσεις εγκεφάλου, η οποία πρόκειται για σημασιολογική τμηματοποίηση
2. Αυτόματη ταξινόμηση άρθρων ειδήσεων, η οποία πρόκειται για επεξεργασία φυσικής γλώσσας (NLP)
3. Αυτόματη επισήμανση προσβλητικών σχολίων σε φόρουμ συζήτησης, η οποία αυτή είναι η ταξινόμηση κειμένου
4. Συνοψίζοντας τα μεγάλα έγγραφα αυτόματα, αυτός είναι ένας κλάδος του NLP που ονομάζεται σύνοψη κειμένου και πάλι χρησιμοποιώντας τα ίδια εργαλεία.
5. Δημιουργία chatbot ή προσωπικού βοηθού, αυτό περιλαμβάνει πολλά στοιχεία NLP.
6. Κάνοντας την εφαρμογή σας να αντιδρά σε φωνητικές εντολές, η οποία πρόκειται για αναγνώριση ομιλίας.
7. Εντοπισμός απάτης με πιστωτικές κάρτες, το οποίο πρόκειται για ανίχνευση ανωμαλιών.
8. Τμηματοποίηση πελατών βάσει των αγορών τους, ώστε να μπορείτε να σχεδιάσετε μια διαφορετική στρατηγική μάρκετινγκ για κάθε τμήμα, αυτό είναι ομαδοποίηση

Τέλος, συνοψίζοντας η μηχανική μάθηση είναι ιδανική για:

Προβλήματα για τα οποία οι υπάρχουσες λύσεις απαιτούν πολλές ρυθμίσεις χειρός ή μεγάλες λίστες κανόνων: ένας αλγόριθμος Machine Learning μπορεί συχνά να απλοποιήσει τον κώδικα και να αποδώσει καλύτερα.

1. Πολύπλοκα προβλήματα για τα οποία δεν υπάρχει καθόλου καλή λύση χρησιμοποιώντας μια παραδοσιακή προσέγγιση: οι καλύτερες τεχνικές μηχανικής

μάθησης μπορούν να βρουν μια λύση.

2. Κυμαινόμενα περιβάλλοντα: ένα σύστημα Machine Learning μπορεί να προσαρμοστεί σε νέα δεδομένα
3. Λήψη πληροφοριών σχετικά με πολύπλοκα προβλήματα και μεγάλες ποσότητες δεδομένων.

1.6. Εισαγωγή στο Matlab

Το Matlab είναι ένα εργαλείο που μας προφέρει ένα διαδραστικό προγραμματιστικό περιβάλλον, το οποίο χρησιμοποιεί έναν αριθμό εφαρμογών. Το Matlab χρησιμοποιεί υψηλού επιπέδου γλώσσα προγραμματισμού η οποία είναι πολύ καλή για την μοντελοποίηση και την επίλυση σύνθετων μαθηματικών προβλημάτων.

Εικόνα 1.6 Matlab

Η ονομασία αυτού το εργαλείου είναι το αρκτικόλεξο από τις λέξεις **Mat**rix **Lab**oratory (εργαστήριο πινάκων) επειδή χρησιμοποιεί αποκλειστικά πίνακες για τη λειτουργία του.

Τα βασικά συστατικά του Matlab είναι:

1. Το Περιβάλλον Ανάπτυξης και τα αντίστοιχα εργαλεία του: περιλαμβάνει διάφορα παράθυρα, όπως το Παράθυρο Εντολών (Command Window) και το Ιστορικό Εντολών (Command History) και άλλα εργαλεία για αποσφαλμάτωση (debugging), ανάλυση κώδικα και πλοήγηση στο σύστημα αρχείων.
2. Η βιβλιοθήκη μαθηματικών συναρτήσεων: ένα από τα πιο σημαντικά συστατικά του Matlab, το οποίο προσφέρει μεγάλο εύρος αριθμητικών συναρτήσεων, από τις πιο απλές μέχρι τις πιο περίπλοκες.
3. Η γλώσσα προγραμματισμού: μια υψηλού επιπέδου προγραμματιστική γλώσσα με δομές δεδομένων, συναρτήσεις, εντολές ελέγχου ροής, εντολές εισόδου/εξόδου και στοιχεία από αντικειμενοστραφείς γλώσσες προγραμματισμού.
4. Τα γραφικά συστατικά: το Matlab παρέχει μια πληθώρα δυνατοτήτων απεικόνισης διανυσμάτων, πινάκων και γραφημάτων στις 2 και 3 διαστάσεις.

Το Matlab μας δίνει τη δυνατότητα να εργαστούμε με δύο τρόπους. Ο πρώτος τρόπος είναι ότι μπορούμε να χρησιμοποιήσουμε μεμονωμένες εντολές στο παράθυρο εντολών και να πάρουμε άμεσα αποτελέσματα είτε μέσω προγραμμάτων τα οποία

γράφονται από το χρήστη στο παράθυρο εισαγωγής κειμένου. Τα προγράμματα αυτά αποθηκεύονται με την κατάληξη .m σαν αρχεία απλού κειμένου και ονομάζονται m-files. Τα ονόματα των m-file δεν επιτρέπεται να περιλαμβάνουν τελείες και συγκεκριμένους ειδικούς χαρακτήρες. Το πρόγραμμα δεν είναι απαραίτητο να περάσει το στάδιο της μεταγλώττισης (compiling), όπως γίνεται σε άλλες γλώσσες προγραμματισμού αλλά γίνεται αυτόματα πριν το στάδιο της εκτέλεσης. Υπάρχουν δυο ειδών τύποι m-files. Τα αρχεία προγραμμάτων (scripts) και τα αρχεία συναρτήσεων(functions). Τα αρχεία προγραμμάτων είναι αυτόνομα, δεν δέχονται ορίσματα εισόδου και δεν επιστρέφουν μεταβλητές εξόδου. Χρησιμοποιούν τις υπάρχουσες μεταβλητές που βρίσκονται στο χώρο εργασίας ή δημιουργούν καινούργιες. Τα αρχεία συναρτήσεων από την άλλη, δέχονται υποχρεωτικά ορίσματα εισόδου και μπορούν να επιστρέψουν μεταβλητές εξόδου. Το όνομα μιας συνάρτησης μαζί με τα ορίσματα εισόδου και τις μεταβλητές εξόδου γράφονται στην πρώτη γραμμή του αντίστοιχου m-file συνοδευόμενα από τη λέξη-κλειδί function, ενώ στην τελευταία γραμμή του αρχείου, γράφεται η λέξη-κλειδί end. Επίσης, το m-file που περιέχει μια συνάρτηση πρέπει να έχει το ίδιο όνομα με αυτή.

Τέλος να αναφέρουμε ότι μια συνάρτηση χρησιμοποιεί το δικό της χώρο μεταβλητών, ο οποίος δεν είναι προσπελάσιμος μετά την εκτέλεσή της. Οι συναρτήσεις μπορούν να κληθούν μέσα από scripts ή από άλλες συναρτήσεις για διάφορες τιμές των ορισμάτων εισόδου τους.

1.7. Τεχνικές Ανίχνευσης Ανθρώπων

Η ανίχνευση των ανθρώπων στις εικόνες είναι ένα δύσκολο έργο σχετικά με την μεταβλητή εμφάνισης τους και το ευρύ φάσμα των θέσεων που μπορούν να υιοθετήσουν. Επίσης η αναγνώριση της παρουσίας ανθρώπου είναι μια από τις πιο σημαντικές λειτουργείες που μπορούν να διεξαχθούν και σε video. Η πρώτη ανάγκη είναι ένα ισχυρό σύνολο χαρακτηριστικών που επιτρέπει την καθαρή διάκριση της ανθρώπινης μορφής. Παρόλα αυτά, αυτή η εργασία γίνεται πιο περίπλοκη με την παρουσία διάφορων μεταβλητών στην εικόνα, όπως είναι η φωτεινότητα, ο φωτισμό, η αντίθεση και το φόντο της εικόνας.

Εικόνα 1.7Δείγμα εικόνας από INRIA

Ύστερα από διάφορες έρευνες που έχουν γίνει πάνω σε αυτό το αντικείμενο καταλήγουμε στο ότι ο πιο κοινός τρόπος για να επιτευχθεί η ανίχνευση ενός ανθρώπου είναι μέσω της ανίχνευσης του προσώπου. Το πρόσωπο του ανθρώπου είναι το πιο ξεχωριστό σημείο πάνω στο ανθρώπινο σώμα, όπου εάν ανιχνευθεί με ακρίβεια οδηγεί σε μια επιτυχή ανίχνευση ανθρώπινης σιλουέτας.

Παρόλα αυτά για την ανίχνευση ανθρώπων σε εικόνες ή video χωρίζονται σε μερικές κατηγορίες ανάλογα με τον τρόπο ανίχνευσης, αυτές οι κατηγορίες είναι η **ανίχνευση προσώπου**, η **ανίχνευση κίνησης**, η **ανίχνευση δέρματος** και ο **εντοπισμός**

χαρακτηριστικών σημείων, όπου θα δούμε παρακάτω κάποια λίγα πράγματα για τον κάθε τρόπο ανίχνευσης.

1. **Ανίχνευση Προσώπου**

Από την δεκαετία του ’70 μέχρι και σήμερα έχουν γίνει πολλές μελέτες πάνω στον τομέα της ανίχνευσης προσώπου. Επίσης έχουν δημοσιευθεί πάρα πολλές έρευνες που σχετίζονται με τους αλγορίθμους που χρησιμοποιούνται σε αυτό τον τομέα. Οι αλγόριθμοι για την ανίχνευση προσώπου μπορούν να χωριστούν σε δύο κατηγορίες. Στην πρώτη κατηγορία ανήκουν οι προσεγγίσεις με βάση τα χαρακτηριστικά (feature-based) όπως για παράδειγμα η “bottom-up” προσέγγιση. Σε αυτή την κατηγορία εξάγονται τα χαρακτηριστικά του προσώπου από μια εικόνα και από αυτά χειριζόμαστε τις παραμέτρους της, όπως είναι οι γωνίες, το μέγεθος και οι αποστάσεις. Στη δεύτερη κατηγορία ανήκουν οι προσεγγίσεις με βάση την εικόνα (image-based), για παράδειγμα προσεγγίσεις με βάση την εμφάνιση (appearancebased) αλλά και η προσέγγιση «template matching». Αυτό το είδος αλγορίθμου, βασίζεται σε τεχνικές εκπαίδευσης μηχανής για να αναγνωρίζει συγκεκριμένα σημεία ενδιαφέροντος.

Παρακάτω θα δούμε ονομαστικά τις πιο κοινές προσεγγίσεις, οι οποίες είναι:

1. Η **Information based (Top-Down)**,
2. Η **Feature Invariant (Bottom-up)** η οποία χωρίζεται σε τρεις κατηγορίες οι οποίες είναι:
   a) Η **Προσέγγιση με βάση το χρώμα ή Μοντέλο δέρματος**,
   b) Η **Προσέγγιση με βάση τα χαρακτηριστικά προσώπου** και
   c) Η **Υφή (Texture)**
3. Η **Template matching methods** η οποία χωρίζεται σε δύο κατηγορίες οι οποίες είναι:
   a) Τα **Προκαθορισμένα Πρότυπα Προσώπου** και
   b) Τα **Παραμορφώσιμα Πρότυπα**
4. Η **Appearance based** η οποία χωρίζεται σε τέσσερις κατηγορίες οι οποίες είναι οι:
   a) **Eigenfaces**
   b) **Distributed-based**
   c) **Neural Network**
   d) **Support Vector Machine**

2. **Ανίχνευση Κίνησης**

Η ανίχνευση κίνησης γίνεται αποκλειστικά μόνο σε video. Στα video έχουμε περισσότερες πληροφορίες σχετικά με το ιστορικό των κινούμενων αντικειμένων, το οποίο βοηθά στο διαχωρισμό του φόντου από το προσκήνιο. Σαν εμφανή κίνηση θεωρούμε την κίνηση που προκύπτει από ένα τυπικό αντικείμενο παρακολούθησης, π.χ. άνθρωπο ή όχημα, σε αντίθεση με τις άλλες παραπλανητικές κινήσεις, όπως είναι οι ακτίνες του φωτός όταν πέφτουν στο νερό ή το ανέμισμα των φύλων ενός δέντρου. Οι παραπλανητικές αυτές κινήσεις, σε ένα πραγματικό περιβάλλον, κάνουν το πρόβλημα της ανίχνευσης κίνησης ακόμη πιο δύσκολο. Γενικότερα, οι κινούμενες περιοχές ανιχνεύονται με τον εντοπισμό των αλλαγών που δημιουργούνται στην ακολουθία εικόνων. Οι περισσότερες εφαρμογές που έχουν δημιουργηθεί πάνω στην ανίχνευση κίνησης, εφαρμόζουν ορισμένα στάδια προ- επεξεργασίας πριν εφαρμόσουν τον αλγόριθμο ανίχνευσης αλλαγών.

3. **Ανίχνευση Δέρματος**

Το χρώμα και η υφή του δέρματος αποτελούν σημαντικές ενδείξεις που οι άνθρωποι χρησιμοποιούν συνειδητά ή ασυνείδητα για να συμπεράνουν μια ποικιλία από πολιτισμικές πτυχές των υπολοίπων. Το χρώμα του δέρματος και η υφή μπορεί να είναι μια ένδειξη της φυλής, της ηλικίας, και της ομορφιάς του καθενός από εμάς, αλλά οι ερμηνείες ποικίλουν ανά τους διαφορετικούς πολιτισμούς και τις χρονικές περιόδους. Στις εικόνες και στα βίντεο όμως, το χρώμα του δέρματος είναι ένα σημάδι παρουσίας ανθρώπου. Η ανίχνευση δέρματος είναι η διαδικασία με την οποία εντοπίζουμε pixel και περιοχές που έχουν το χρώμα του δέρματος. Αυτή η διαδικασία χρησιμοποιείται συνήθως ως ένα βήμα προεπεξεργασίας για να βρούμε τις περιοχές που περιέχουν ενδεχομένως ανθρώπινα πρόσωπα και άκρα σε εικόνες.

Η διαδικασία της ανίχνευσης δέρματος έχει δυο φάσεις. Τη φάση της εκμάθησης, και τη φάση της ανίχνευσης. Το στάδιο της εκπαίδευσης περιλαμβάνει τρία βασικά στάδια:

a) Συλλογή μιας βάσης δεδομένων με τόνους δέρματος από διαφορετικές εικόνες. Μια τέτοια βάση συνήθως περιέχει τόνους από μια μεγάλη ποικιλία ανθρώπων σε διαφορετικές καταστάσεις φωτός.

b) Να διαλέξουμε τον κατάλληλο χώρο χρώματος.

c) Εκμάθηση των παραμέτρων ενός ταξινομητή δέρματος .

Σε κάθε εκπαιδευόμενο ανιχνευτή δέρματος, ο προσδιορισμός των pixel του δέρματος γίνεται σε τρείς φάσεις:

a) Τη μετατροπή της εικόνας στον ίδιο χώρο χρωμάτων που χρησιμοποιήσαμε στο στάδιο της εκμάθησης.

b) Να κατατάξουμε τα pixel με τη χρήση του ταξινομητή δέρματος σε δέρμα ή μη-δέρμα.

c) Τέλος, τα pixel επεξεργάζονται μορφολογικά για να επιβληθεί χωρική ομοιογένεια στις περιοχές που εντοπίστηκαν.

4. **Εντοπισμός χαρακτηριστικών Σημείων**

Έπειτα από διάφορα πειράματα που έκανε ο Johansson το ’70 αποδείχθηκε ότι αρκεί μόνο η κίνηση των άκρων προκειμένου να αναγνωριστεί ανθρώπινη παρουσία από κάποιον. Δεδομένου ότι ο άνθρωπος μπορεί χωρίς προσπάθεια να αναγνωρίσει την κίνηση, είναι πιθανό και ένας αλγόριθμος να κάνει το ίδιο. Το πείραμα του Johansson μας δείχνει ότι η ανίχνευση ανθρώπου από την κίνηση των χαρακτηριστικών αυτών σημείων είναι ένας ρεαλιστικός στόχος. Με αυτό το σκεπτικό στο μυαλό μας, μπορούμε να χωρίσουμε ένα τέτοιο σύστημα σε 3 διαφορετικές ρουτίνες.

1. Εξαγωγή των χαρακτηριστικών σημείων από βίντεο.
2. Εύρεση ενός μοντέλου που αντιπροσωπεύει με ακρίβεια την ανθρώπινη κίνηση.
3. Εφαρμογή αυτού του μοντέλου σε έναν ανιχνευτή, ο οποίος θα αποφασίζει αν μια ομάδα από κινούμενα σημεία είναι αντιπροσωπευτική ανθρώπινης κίνησης.

Κεφάλαιο 2

2.1. HOG (Histogram of Oriented Gradients)

2.1.1. Εισαγωγή

Το HOG είναι ένας περιγραφέας χαρακτηριστικών που έχει χρησιμοποιηθεί ευρέως και επιτυχώς για την ανίχνευση αντικειμένων. Το HOG πρώτο χρησιμοποιήθηκε το 1994 από το Mitsubishi Electric Research Laboratories, αλλά έγινε γνωστό το 2005 από τους Navneet Dalal και Bill Triggs ερευνητές του γαλλικού Εθνικού Ινστιτούτου Έρευνας Πληροφορικής και Αυτοματισμού (INRIA) όταν παρουσίασαν τις συμπληρωματικές εργασίες τους για την ανίχνευση πεζών σε στατικές εικόνες.

Εικόνα 2.1 HOG

Αντιπροσωπεύει αντικείμενα ως ένα φορέα χαρακτηριστικών σε αντίθεση με ένα σύνολο διανυσμάτων χαρακτηριστικών όπου το καθένα αντιπροσωπεύει ένα τμήμα ή την εικόνα. Το HOG ως ένας feature descriptor ο οποίος κατά το πλείστων χρησιμοποιείται για την εξαγωγή στοιχείων από δεδομένα εικόνων.

Ο feature descriptor με απλά λόγια θα μπορούσαμε να πούμε ότι μπορεί να βρίσκεις τις πληροφορίες μιας εικόνας, δηλαδή εάν σε μια εικόνα αυτό που βλέπουμε είναι ένας σκύλος ή ένα αυτοκίνητο μαζί και με ότι άλλες πληροφορίες μπορεί να περιέχει αυτή η εικόνα. Υπολογίζεται με συρόμενο ανιχνευτή παραθύρου πάνω από μια εικόνα ,όπου ένας περιγραφέας HOG είναι υπολογισμένος για κάθε θέση. Όπως το SIFT, η κλίμακα της εικόνας προσαρμόζεται (πυραμίδα).

Με λίγα λόγια ένας feature descriptor είναι μια απλοποιημένη προβολή της εικόνας που περιέχει τις πιο απαραίτητες πληροφορίες σχετικά με την εικόνα. Εκτός από τον HOG άλλοι δημοφιλείς feature descriptor είναι ο SIFT (Scale Invariant Feature Transform) και ο SURF (Speed-Up Robust Feature). Τα HOGS χρησιμοποιούνται συχνά με ταξινομητές SVM. Κάθε περιγραφέας HOG ότι υπολογίζεται τροφοδοτείται σε ένα ταξινομητή SVM για να προσδιορίσει εάν το αντικείμενο βρέθηκε ή όχι.

2.1.2. Τρόπος Λειτουργίας

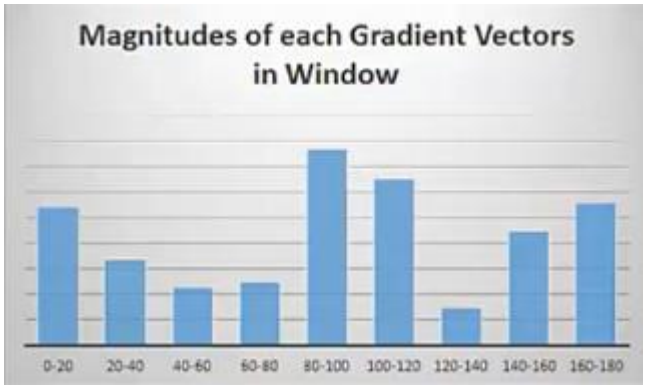
Όπως είδαμε παραπάνω το HOG είναι ένας περιγραφέας χαρακτηριστικών ο οποίος υπολογίζεται με συρόμενο ανιχνευτή παραθύρου. Μετά από όλα αυτά θα δούμε όσο το δυνατόν πιο λεπτομερώς βήμα βήμα τον τρόπο λειτουργίας του.

Στην αρχή ξεκινάμε χρησιμοποιώντας ένα παράθυρο ή κελί ανίχνευσης 8x8 εικονοστοιχείων και υπολογίζουμε τις διανύσεις διαβάθμισης διανύσματος ή ακμών σε κάθε εικονοστοιχείο. Έπειτα αυτό δημιουργεί διανύσματα διαβάθμισης 64 (8x8) τα

οποία στη συνέχεια αντιπροσωπεύονται ως ιστόγραμμα.

Στη συνέχεια, κάθε κελί χωρίζεται σε γωνιακούς κάδους, όπου κάθε κάδος αντιστοιχεί σε μια κατεύθυνση του βηματικού (π.χ. X.Y). Σε χαρτί Dalne και Triggs χρησιμοποίησαν 9 κάδους 0-180^{o} (20^{o} κάθε κάδο). Αυτό επαναφέρει αποτελεσματικά τα διανύσματα σε μόλις 9 τιμές. Δεδομένου ότι αποθηκεύει τα μεγέθη των βαθμίδων, είναι σχετικά άνοσο στις παραμορφώσεις.

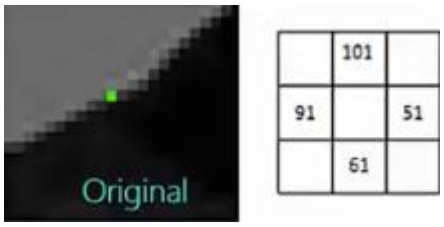
Εικόνα 2.2Παράδειγμα εικόνας για HOG



.

Έπειτα από όλα αυτά ομαλοποιούμε τις κλίσεις για να διασφαλίσουμε την αμετάβλητη αλλαγή του φωτισμού, δηλαδή τη φωτεινότητα και την αντίθεση.

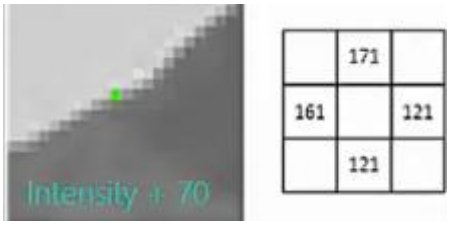
π.χ. στις παρακάτω εικόνες αν βυθίσουμε τα διανύσματα από τα μεγέθη κλίσης έχουμε 0.707 για όλους, αυτό είναι η κανονικοποίηση.



$$\Delta_{\mathrm{H}} = 50$$

$$\Delta_{\mathrm{V}} = 50$$

$$|\Delta| = \sqrt{50^2 + 50^2} = 70.72$$



$$\Delta_H = 50$$

$$\Delta_V = 50$$

$$|\Delta| = \sqrt{50^2 + 50^2} = 70.72$$

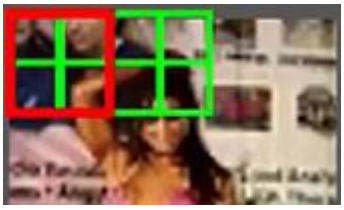
	181	
171		71
	81	

$$\Delta_H = 100$$

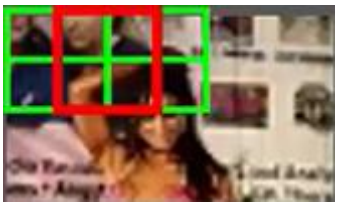
$$\Delta_V = 100$$

$$|\Delta| = \sqrt{100^2 + 100^2} = 141.42$$

Αντί για κανονικοποίηση μεμονωμένων παραθύρων, χρησιμοποιείται μια μέθοδος που ονομάζεται Block Normalization. Αυτό λαμβάνει υπόψη τα γειτονικά μπλοκ, έτσι ομαλοποιούμε λαμβάνοντας υπόψη μεγαλύτερα τμήματα της εικόνας.



Εικόνα 2.2.1



Εικόνα 2.2.2

2.2. PCA (Principal Component Analysis)

2.2.1. Εισαγωγή

Το PCA εφευρέθηκε το 1901 από τον Karl Pearson, ως ανάλογο του θεωρήματος κύριου άξονα στη μηχανική. Το PCA μπορεί να θεωρηθεί ως προσαρμογή ενός ελλειψοειδούς διαστάσεων σε δεδομένα, όπου κάθε άξονας του ελλειψοειδούς αντιπροσωπεύει ένα κύριο συστατικό. Εάν κάποιος άξονας του ελλειψοειδούς είναι μικρός, τότε η διακύμανση κατά μήκος αυτού του άξονα είναι επίσης μικρή και παραλείποντας αυτόν τον άξονα και το αντίστοιχο κύριο συστατικό του από την αναπαράσταση του συνόλου δεδομένων, χάνουμε μόνο μια εξίσου μικρή ποσότητα πληροφοριών.

Επίσης το PCA ορίζεται ως ένας ορθογώνιος γραμμικός μετασχηματισμός που μετατρέπει τα δεδομένα σε ένα νέο σύστημα συντεταγμένων έτσι ώστε η μεγαλύτερη

διακύμανση με κάποια κλιματική προβολή των δεδομένων να βρίσκεται στην πρώτη συντεταγμένη, η δεύτερη μεγαλύτερη διακύμανση στο δεύτερη συντεταγμένη και ούτω καθεξής. Το PCA χρησιμοποιείται κυρίως ως εργαλείο στην διερευνητική ανάλυση δεδομένων και για τη δημιουργία προγνωστικών μοντέλων. Χρησιμοποιείται συχνά για να απεικονίσει τη γενετική απόσταση και τη σχέση μεταξύ των πληθυσμών.

Το PCA γίνεται είτε με αποσύνθεση μοναδικής τιμής ενός σχεδιαστικού πίνακα είτε με τα ακόλουθα 2 βήματα:

1. Υπολογισμός της μήτρας συνδιακύμανσης δεδομένων των αρχικών δεδομένων.
2. Εκτέλεση αποσύνθεσης ιδιοτιμής στην μήτρα συνδιακύμανσης.

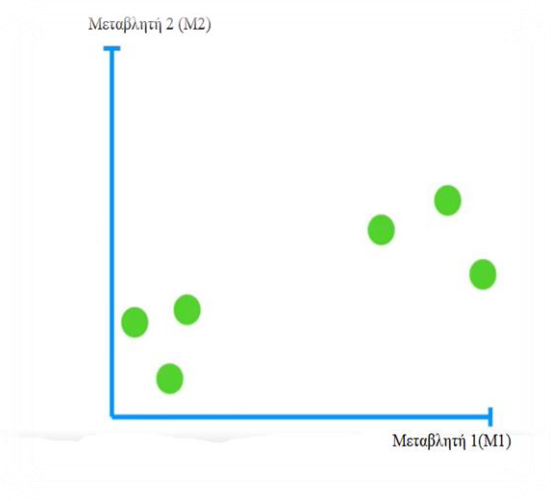
Ακόμα θα μπορούσαμε να πούμε ότι το PCA είναι η απλούστερη από τις πραγματικές πολυπαραγοντικές αναλύσεις που βασίζονται σε ιδιοδιανύσματα. Συχνά, η λειτουργία του μπορεί να θεωρηθεί ότι αποκαλύπτει την εσωτερική δομή των δεδομένων με τρόπο που εξηγεί καλύτερα τη διακύμανση στα δεδομένα. Εάν ένα σύνολο δεδομένων πολλαπλών παραλλαγών εμφανίζεται ως σύνολο συντεταγμένων σε ένα χώρο δεδομένων υψηλής διαστάσεων, το PCA μπορεί να παρέχει στον χρήστη μια εικόνα χαμηλότερης διάστασης, μια προβολή αυτού του αντικειμένου όταν προβάλλεται από την πιο ενημερωτική του άποψη. Αυτό γίνεται χρησιμοποιώντας μόνο τα πρώτα λίγα βασικά στοιχεία έτσι ώστε να μειωθεί η διάσταση των μετασχηματισμένων δεδομένων.

Επιπλέον το PCA σχετίζεται στενά με την ανάλυση παραγόντων. Η ανάλυση παραγόντων ενσωματώνει τυπικά περισσότερες ειδικές παραμέτρους για τον τομέα σχετικά με την υποκείμενη δομή και επιλύει ιδιοδιανύσματα ενός ελαφρώς διαφορετικού πίνακα.

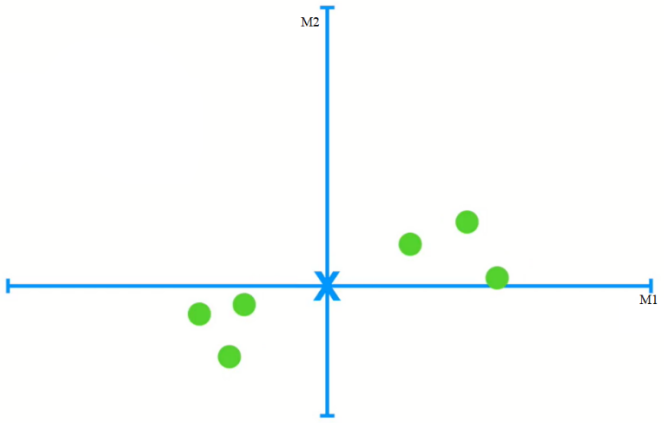
Τέλος το PCA σχετίζεται επίσης με ανάλυση κανονικής συσχέτισης (CCA). Το CCA ορίζει συστήματα συντεταγμένων που περιγράφουν βέλτιστα τη διασταυρούμενη συνδιακύμανση μεταξύ δύο συνόλων δεδομένων, ενώ το PCA ορίζει ένα νέο ορθογώνιο σύστημα συντεταγμένων που περιγράφει βέλτιστα τη διακύμανση σε ένα μόνο σύνολο δεδομένων.

2.2.2.Τρόπος Λειτουργίας

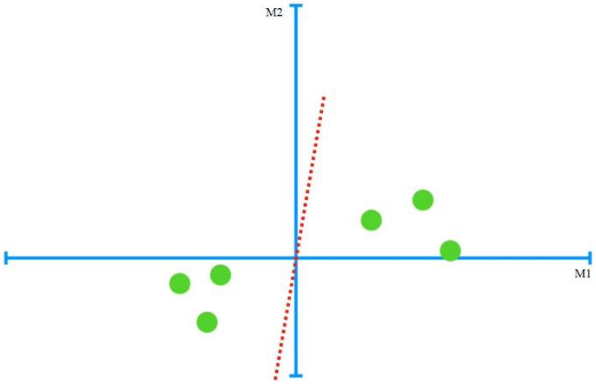
Ο τρόπος λειτουργίας του αλγόριθμου PCA γίνεται πάνω σε ένα καρτεσιανό άξονα όπου τοποθετούμε μέσα στα πλαίσια αυτού τις τιμές μας.



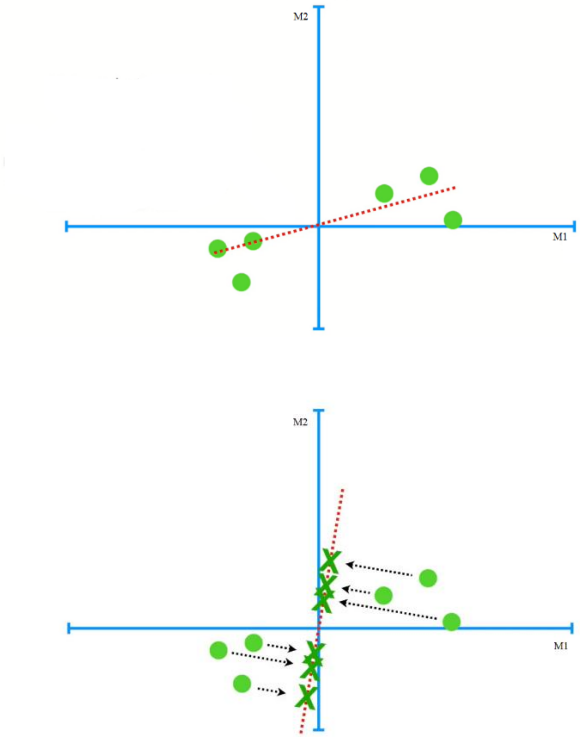
Έπειτα μετακινούμε τις τιμές μας έτσι ώστε να είναι κοντά στο κέντρο των αξόνων, βέβαια χωρίς να έχουν αλλάξει η μεταξύ τους απόσταση.



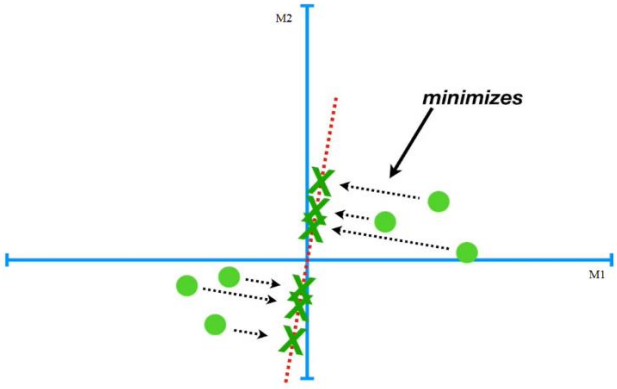
Στη συνέχεια φτιάχνουμε μια τυχαία ευθεία γραμμή η οποία θα περνάει από το κέντρο των αξόνων.



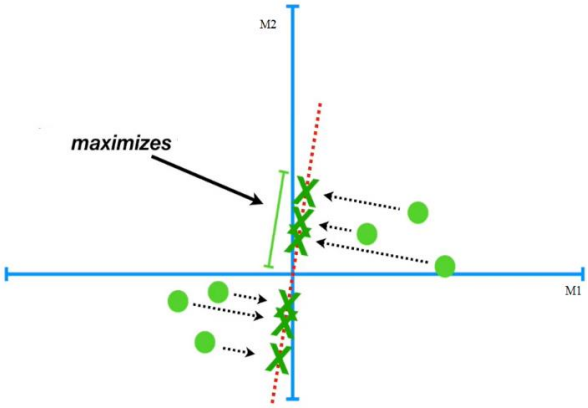
Καθώς έχουμε σχεδιάσει τη γραμμή μας ξεκινάμε να την γυρνάμε έτσι ώστε να εφαρμόζεται καλύτερα στα δεδομένα μας.



Για να πούμε ότι η γραμμή εφαρμόζει καλύτερα στα δεδομένα μας πρέπει να ελαχιστοποιήσουμε συνολικά (όσο το δυνατόν πιο πολύ) τις αποστάσεις που έχουν τα σημεία μας με τη γραμμή,

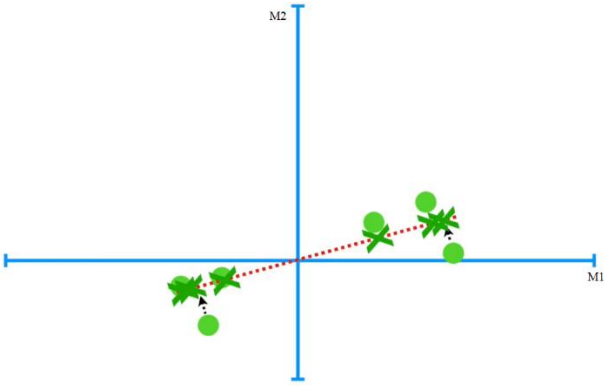


ή να μεγιστοποιήσουμε την απόσταση των σημείων (που είναι πάνω στη γραμμή) από το κέντρο των αξόνων.



Ένας τρόπος για να έχουμε μεγαλύτερη ακρίβεια στον υπολογισμό για την καταλληλότητα της γραμμής, αθροίζουμε τα τετράγωνα των αποστάσεων που είναι πάνω στη γραμμή, αυτό το κάνουμε καθώς μετακινούμε τη γραμμή, το μεγαλύτερο αποτέλεσμα (SS(distances)) είναι και η καλύτερη επιλογή. Η γραμμή που βρήκαμε ότι είναι η καλύτερη επιλογή ονομάζεται

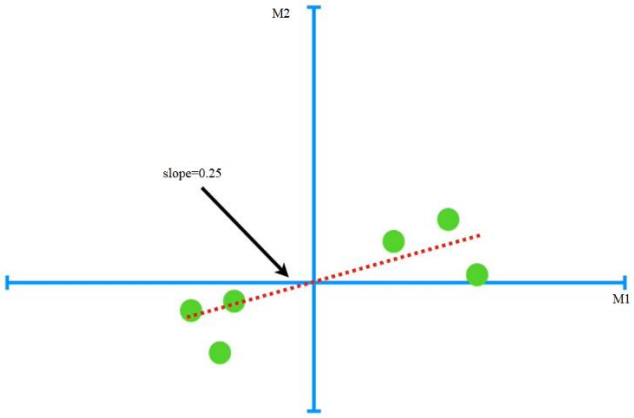
Principal Component (**PC**).



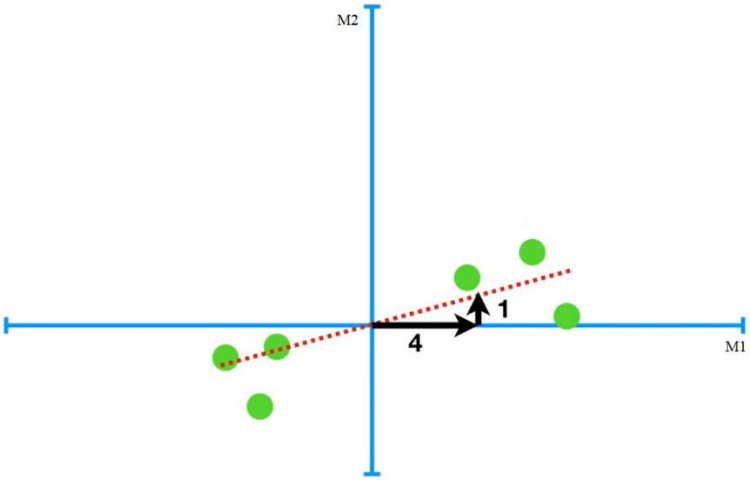
Έπειτα βρίσκουμε την κλίση (slope) της γραμμής μας και από εκεί πόσα κομμάτια (parts) παίρνουμε από την κάθε μεταβλητή μας. Για την καλύτερη κατανόηση θα προχωρήσουμε σε ένα παράδειγμα.

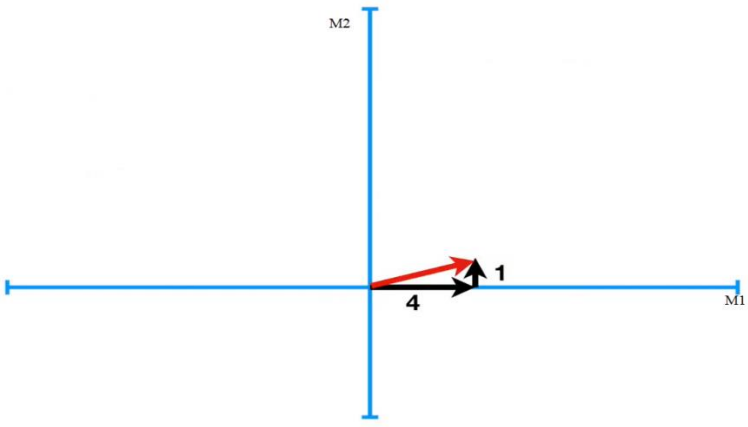
Principal Component 1 (**PC1**):

Έστω ότι το slope = 0.25



για να φτιάξουμε το PC1 παίρνουμε mix 4 parts από την μεταβλητή 1 και 1 part από την μεταβλητή 2





$$a^2 = b^2 + c^2 = 4^2 + 1^2 = 16 + 1 \Rightarrow$$

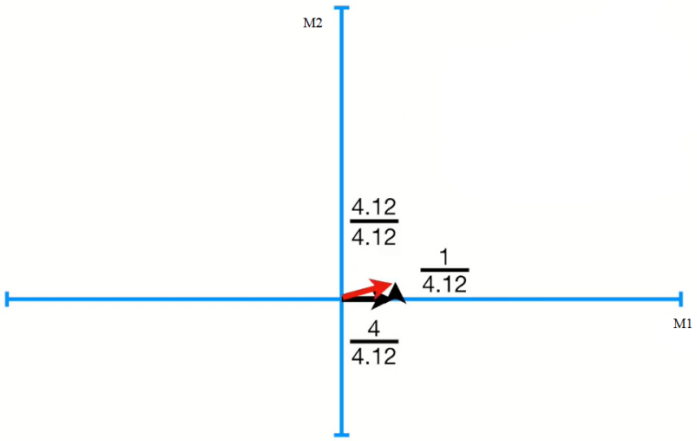
$$a^2 = 17 \Rightarrow a = \sqrt{17} \Rightarrow$$

$$a = 4.12$$

Αφού βρούμε την υποτείνουσα μετατρέπουμε τις αποστάσεις έτσι ώστε η υποτείνουσα να ισούται με 1 (4.12 / 4.12)

$$\frac{4.12}{4.12} = \frac{\sqrt{4^2 + 1^2}}{4.12} = \sqrt{\frac{4^2 + 1^2}{4.12^2}} = \sqrt{\frac{4}{4.12}^2 + \frac{1}{4.12}^2} \Rightarrow a = 1, b = 0.97, c = 0.242$$

Οπότε με τα τελευταία δεδομένα το σχήμα μας γίνεται mix 0.97 parts από την μεταβλητή 1 και 0.242 parts από την μεταβλητή 2

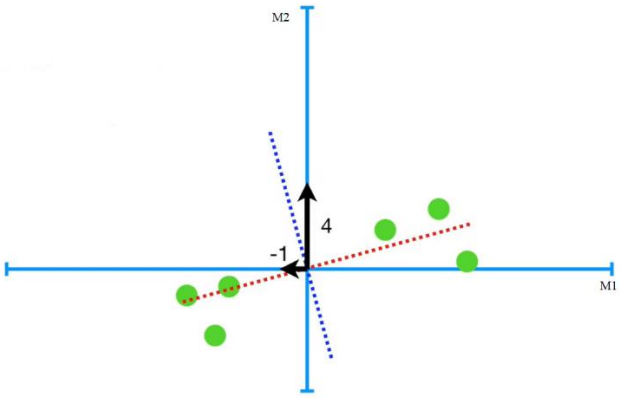


$$SS(distances\ for\ PC1) = Eigenvalue\ for\ PC1$$

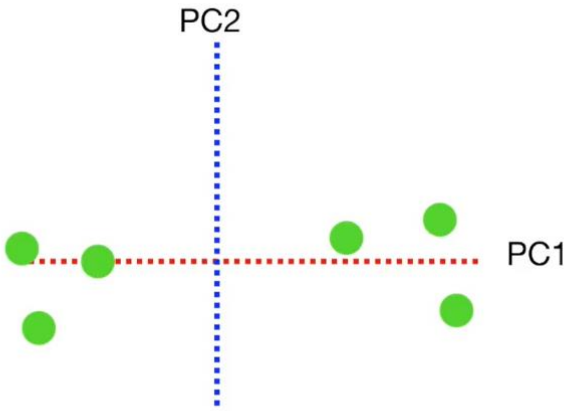
$$\sqrt{Eigenvalue\ for\ PC1} = Signular\ Value\ for\ PC1$$

Principal Component 2 (PC2)

Για τον υπολογισμό του PC2 ενεργούμε ακριβώς με όμοιο τρόπο όπως και στο PC1. Παίρνουμε τον άξονα μαζί με τα σημεία μας και την καλύτερη επιλογή γραμμής από το PC1 και ξεκινάμε να βρούμε την καλύτερη επιλογή για το PC2, οπότε καταλήγουμε να πάρουμε -1 parts από την μεταβλητή 1 και 4 parts από την μεταβλητή 2



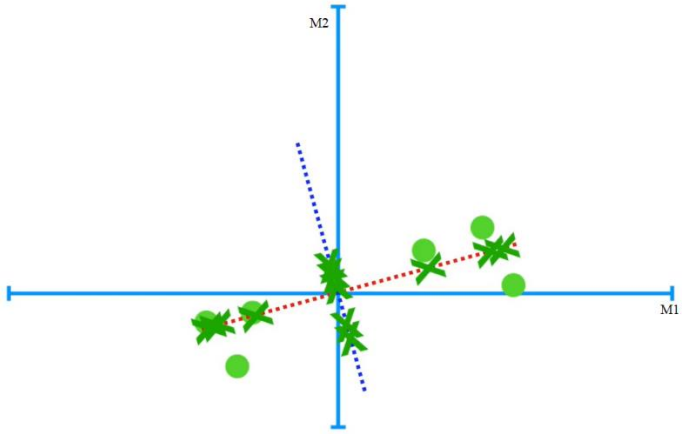
Με τους ίδιους υπολογισμούς όπως κάναμε και στο PC1 κάνουμε κι εδώ φτάνοντας στο σημείο να έχουμε -0.242 parts στη μεταβλητή 1 και 0.97 parts στη μεταβλητή 2.



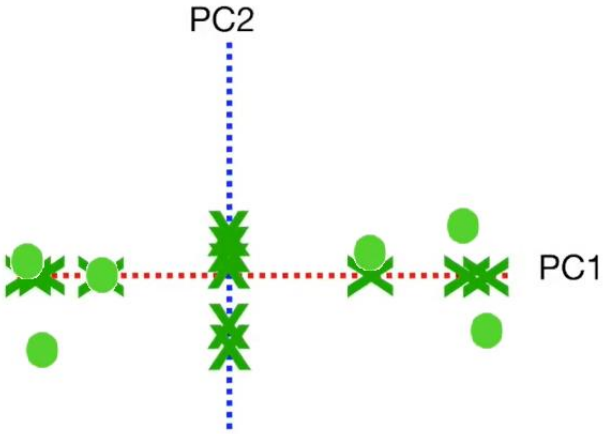
$$d_1^2 + d_2^2 + d_3^2 + \cdots + d_n^2 = SS(distances)$$

$$SS(distances\ for\ PC2) = Eigevalue\ for\ PC2$$

Στην συνέχεια έχουμε τον άξονα με τις καλύτερες επιλογές από το PC1 και το PC2 και σχεδιάζουμε πάνω στις γραμμές τα σημεία.



Έπειτα παίρνουμε τις καλύτερες επιλογές και τα σημεία και τα φέρνουμε σε κάθετη θέση



(Τα σημεία πάνω στις γραμμές είναι με βάση πριν την μετακίνηση)

Αυτή η διαδικασία ονομάζεται Singular Value Decomposition (SVD)

$$SS(distances\ for\ PC1) = Eigenvalue\ for\ PC1$$

$$SS(distances\ for\ PC2) = Eigenvalue\ for\ PC2$$

$$\frac{SS(distances\ for\ PC1)}{n-1} = Variation\ for\ PC1$$

$$\frac{SS(distances\ for\ PC2)}{n-1} = Variation\ for\ PC2$$

π.χ. Variation

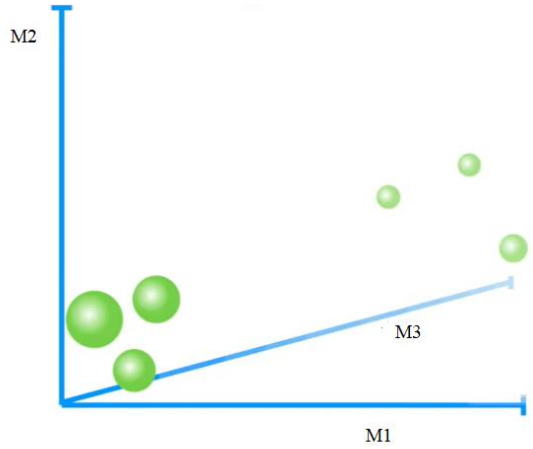
$$PC1 = 15\ και\ PC2 = 3\ \Rightarrow$$

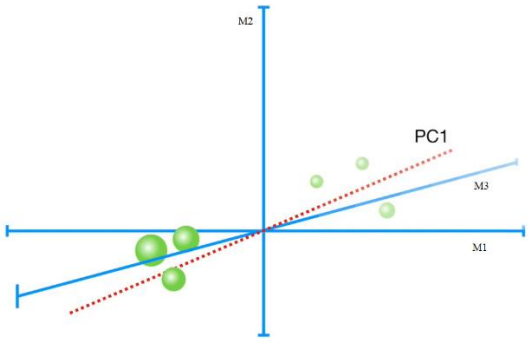
$$PCs = 15 + 3 = 18\ \Rightarrow$$

$$PC1 = {}^{15}/_{18} = 0.83 = 83\%\ \&\ PC2 = {}^{3}/_{18} = 0.17 = 17\%$$

Principal Component 3 (PC3)

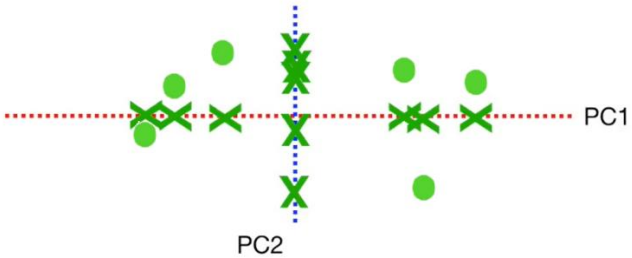
Έστω ότι έχουμε τρεις μεταβλητές, οπότε στο PC1 παίρνουμε 0.62 parts από την μεταβλητή 1 0.15 parts από την μεταβλητή 2 και 0.77 parts από την μεταβλητή 3 και στο PC2 παίρνουμε 0.77 parts από την μεταβλητή 1 0.62 parts από την μεταβλητή 2 και 0.15 parts από την μεταβλητή 3.





PC1+PC3=94%

Από 3D το κάνουμε 2D.



Εάν στο PC3 τα ποσοστά μας (%) είναι όλα κοντινές τιμές (π.χ. PC1=0.77, PC2=0.62, PC3=0.50) και πάρουμε τα 2 πρώτα PC2 τότε δεν θα έχουμε ακριβείς δεδομένα.

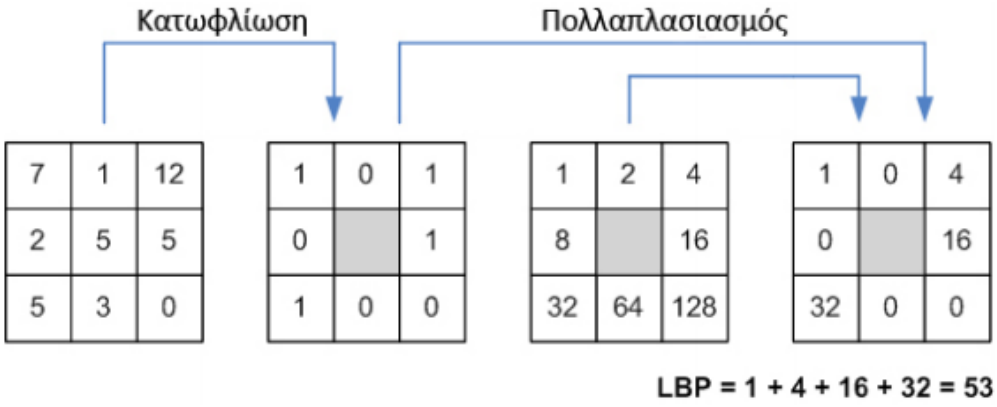
2.3. LBP (Local Binary Patterns)

2.3.1. Εισαγωγή

Το LBP είναι ένας τύπος αλγορίθμου οπτικού περιγραφέα για ταξινόμηση, είναι ένας απλός, αλλά πολύ αποτελεσματικός τελεστής αναγνώρισης υφής που χαρακτηρίζει τα pixels μιας εικόνας ελέγχοντας τη γειτονιά γύρω από κάθε pixel και μετατρέποντας το αποτέλεσμα σε ένα δυαδικό αριθμό. Ο LBP λόγω της εύκολης χρήσης του έχει γίνει πολύ δημοφιλής, ιδίως σε πραγματικές εφαρμογές. Επιπλέον ένας ακόμα λόγος που είναι ευρέως γνωστό είναι για την χαμηλή υπολογιστή πολυπλοκότητα που δίνει τη δυνατότητα ανάλυσης εικόνων σε απαιτητικές εφαρμογές πραγματικού χρόνου. Επίσης μπορεί να χρησιμοποιηθεί για αναγνώριση προσώπου ή στην δικιά μας περίπτωση για αναγνώριση ανθρώπων.

2.3.2. Τρόπος Λειτουργίας

Για παράδειγμα σε κάθε 3x3 block συγκρίνουμε το κεντρικό pixel με τα υπόλοιπα 8 γειτονικά pixel. Εάν ο γείτονας είναι μικρότερος του κεντρικού τότε ο γείτονας γίνεται 0. Εάν ο γείτονας είναι μεγαλύτερος ή ίσος του κεντρικού τότε ο γείτονας γίνεται 1. Οι τιμές έντασης στην ελεγχόμενη γειτονιά πολλαπλασιάζονται με ένα συντελεστή βάρους ίσο με μια δύναμη του 2, που αντιστοιχεί σε κάθε pixel ανάλογα με τη θέση του, όπως φαίνεται στην παρακάτω εικόνα. Η σειρά ανάθεσης των βαρών στα γειτονικά pixel δεν υπακούει κάποιο ιδιαίτερο κανόνα, αλλά ως συνήθως είναι στην ευχέρεια του χρήστη να την επιλέξει, το οποίο αποτελεί ένα μειονέκτημα της μεθόδου, αφού διαφορετικά βάρη θα δώσουν διαφορετικό αριθμό LBP για την ίδια περιοχή. Τέλος αθροίζονται τα αποτελέσματα δίνοντας έτσι τον αριθμό LBP της γειτονιάς. Όπως είναι φυσικό, σε μια γειτονιά 3 × 3 μπορεί να προκύψουν 28 = 256 διαφορετικοί συνδυασμοί περιγραφής υφής.

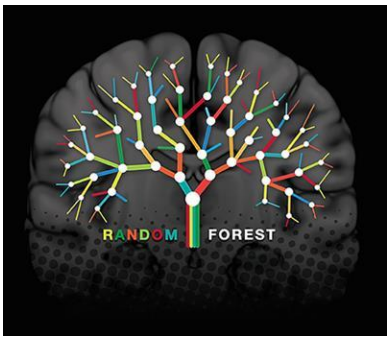


Εικόνα 2.3 Τρόπος υπολογισμού LBP σε γειτονιά 3X3

Κεφάλαιο 3

3.1. Random Forest

3.1.1. Εισαγωγή



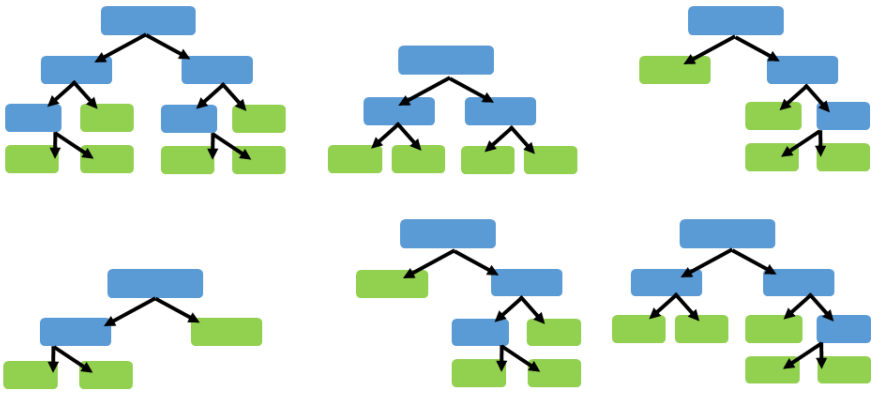
Εικόνα 3.1 Random Forest

Το Random Forest είναι μια μέθοδος μάθησης για την ταξινόμηση. Το Random Forest είναι φτιαγμένο από δέντρα αποφάσεων. Στην απόφαση μηχανικής μάθησης τα δέντρα είναι μια τεχνική για την δημιουργία προγνωστικών μοντέλων. Ονομάζονται δέντρα αποφάσεων επειδή η πρόβλεψη ακολουθεί διάφορους κλάδους του «εάν τότε…» διαχωρίζεται η απόφαση, παρόμοια με τα κλαδιά ενός δέντρου. Αν φανταζόμαστε ότι ξεκινάμε με ένα δείγμα, για το οποίο θέλουμε να προβλέψουμε μια τάξη, θα ξεκινήσουμε από το κάτω μέρος ενός δέντρου και θα ταξιδέψουμε μέχρι τον κορμό μέχρι να φτάσουμε στον πρώτο κλάδο. Αυτή η διάσπαση μπορεί να θεωρηθεί ως χαρακτηριστικό της μηχανικής μάθησης, ας πούμε ότι θα ήταν «ηλικία», θα κάναμε τώρα μια απόφαση σχετικά με το ποιο υποκατάστημα θα ακολουθήσει: "εάν το δείγμα μας έχει ηλικία μεγαλύτερη από 30, συνεχίστε κατά μήκος του αριστερού κλάδου, αλλιώς συνεχίστε κατά μήκος του δεξιού κλάδου". Αυτό θα κάνουμε μέχρι να έρθουμε στον επόμενο κλάδο και να επαναλάβουμε την ίδια διαδικασία λήψης αποφάσεων έως ότου δεν υπάρχουν άλλα υποκαταστήματα μπροστά μας. Αυτό το τελικό σημείο ονομάζεται φύλλο κι στην απόφαση τα δέντρα θα αντιπροσωπεύουν το τελικό αποτέλεσμα: μια προβλεπόμενη τάξη ή τιμή. Σε κάθε κλάδο, βρίσκεται το κατώτατο όριο χαρακτηριστικών που χωρίζει καλύτερα τα (εναπομείναντα) δείγματα τοπικά. Οι πιο συνηθισμένες μετρήσεις για τον ορισμό του «καλύτερου διαχωρισμού» είναι η ακαθαρσία gini και το κέρδος πληροφοριών για εργασίες ταξινόμησης και μείωση διακύμανσης για παλινδρόμηση. Τα δέντρα μεμονωμένων αποφάσεων είναι πολύ εύκολο να οπτικοποιηθούν και να κατανοηθούν επειδή ακολουθούν μια μέθοδο λήψης αποφάσεων που μοιάζει πολύ με τον τρόπο με τον οποίο εμείς οι άνθρωποι λαμβάνουμε αποφάσεις: με μια αλυσίδα απλών κανόνων.

Ωστόσο, δεν είναι πολύ ανθεκτικά, δηλαδή δεν γενικεύονται καλά σε αόρατα δείγματα. Εδώ παίζουν τα τυχαία δάση. Τα δέντρα αποφάσεων λειτουργούν τέλεια με τα δεδομένα που χρησιμοποιήθηκαν για τη δημιουργία τους αλλά δεν προσαρμόζονται εύκολα όταν έρχονται σε επαφή με καινούργια δείγματα. Ο πρώτος αλγόριθμος για το Random Forest δημιουργήθηκε από τον Tin Kam Ho το 1995 χρησιμοποιώντας την μέθοδο της τυχαίας υποχώρο, η οποία είναι ένας τρόπος για την εφαρμογή της προσέγγισης "στοχαστική διάκριση" με την ταξινόμηση που προτείνει ο Eugene Kleinberg. Ο Tin Kam Ho διαπίστωσε ότι τα δάση των δέντρων που χωρίζονται με πλάγια υπερπλάνα μπορούν να αποκτήσουν ακρίβεια καθώς και μεγαλώνουν χωρίς να υποφέρουν από υπερβολική προπόνηση, αρκεί τα δάση να είναι τυχαία περιορισμένα ώστε να είναι ευαίσθητα μόνο σε επιλεγμένες διαστάσεις χαρακτηριστικών.

3.1.2. Τρόπος Λειτουργίας

Για να δημιουργήσουμε ένα Random Forest πρέπει πρώτα να φτιάξουμε μια bootstrapped βάση δεδομένων από την original βάση δεδομένων. Η bootstrapped βάση δεδομένων πρέπει να έχει το ίδιο μέγεθος με την original. Για τη δημιουργία της bootstrapped βάση δεδομένων παίρνουμε τυχαία δεδομένα με τυχαία σειρά από την original βάση δεδομένων, ένα δεδομένο μπορούμε να το πάρουμε πάνω από μια φορά. Έπειτα για την δημιουργία του Random Forest δημιουργούμε ένα δέντρο αποφάσεων χρησιμοποιώντας την bootstrapped βάση δεδομένων αλλά χρησιμοποιώντας μόνο ένα τυχαίο αντικείμενο από κάθε μεταβλητή σε κάθε βήμα. Επαναλαμβάνοντας αυτή τη διαδικασία από την αρχή ξανά και ξανά φτιάχνοντας κάθε φορά κι ένα διαφορετικό δέντρο αποφάσεων.



Εικόνα 3.2 Δέντρα Αποφάσεων στο Random Forest

Έχοντας φτιάξει το Random Forest για να το χρησιμοποιήσουμε παίρνουμε μια καινούργια περίπτωση και την εφαρμόζουμε στα δέντρα αποφάσεων που φτιάξαμε. Έπειτα κρατάμε τις τελικές αποφάσεις στο ερώτημα μας και βλέπουμε πόσες θετικές και αρνητικές αποφάσεις πήραμε, όσες είναι οι περισσότερες αυτή είναι και η τελική απάντηση σε αυτή την περίπτωση.

3.2. SVM (Support Vector Machine)

3.2.1. Εισαγωγή

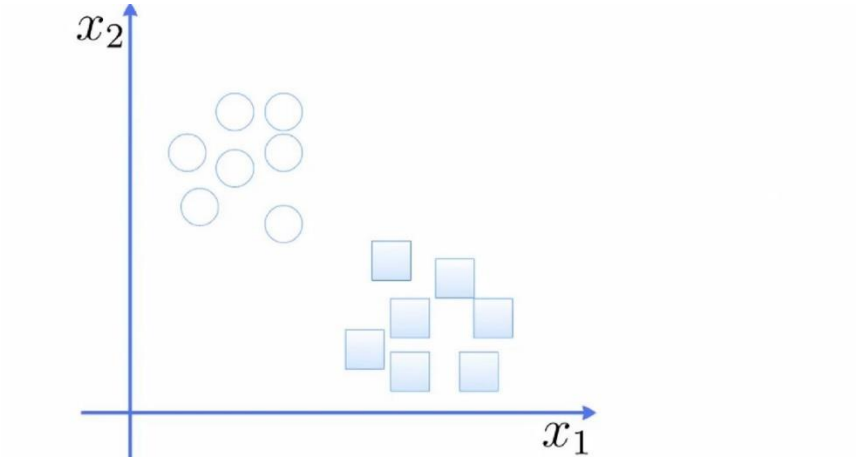
Ο Support Vector Machine (SVM) είναι ένα γραμμικό μοντέλο για προβλήματα ταξινόμησης και παλινδρόμησης. Μπορεί να λύσει γραμμικά και μη γραμμικά προβλήματα και να λειτουργήσει καλά για πολλά πρακτικά προβλήματα. Χρησιμοποιεί μια τεχνική που ονομάζεται kernel trick για να μετασχηματίσει τα δεδομένα μας και στη συνέχεια με βάση αυτούς τους μετασχηματισμούς βρίσκει ένα βέλτιστο όριο ανάμεσα στις πιθανές εξόδους. Ο αρχικός αλγόριθμος SVM επινοήθηκε από τους Vladimir N. Vapnik και Alexey Ya. Chervonenkis το 1963. Το 1992, οι Bernhard E. Boser, Isabelle M. Guyon και Vladimir N. Vapnik πρότειναν έναν τρόπο δημιουργίας μη γραμμικών ταξινομητών εφαρμόζοντας το τέχνασμα του

πυρήνα σε υπερβολικά περιθώρια μέγιστου περιθωρίου. Ένας πιο επίσημος ορισμός για τον SVM είναι ότι, ένα μηχάνημα φορέα υποστήριξης κατασκευάζει ένα υπερπλάνο ή ένα σύνολο υπερπλάνων σε ένα χώρο υψηλού ή άπειρου διαστάσεων, το οποίο μπορεί να χρησιμοποιηθεί για ταξινόμηση , παλινδρόμηση ή άλλες εργασίες όπως ανίχνευση ακραίων τιμών. Διαισθητικά, επιτυγχάνεται ένας καλός διαχωρισμός από το υπερπλάνο που έχει τη μεγαλύτερη απόσταση από το πλησιέστερο σημείο εκπαίδευσης-δεδομένων οποιασδήποτε κλάσης (το λεγόμενο λειτουργικό περιθώριο), καθώς γενικά όσο μεγαλύτερο είναι το περιθώριο, τόσο χαμηλότερο είναι το σφάλμα γενίκευσης του ο ταξινομητής. Τα SVM μπορούν να χρησιμοποιηθούν για την επίλυση διαφόρων πραγματικών προβλημάτων:

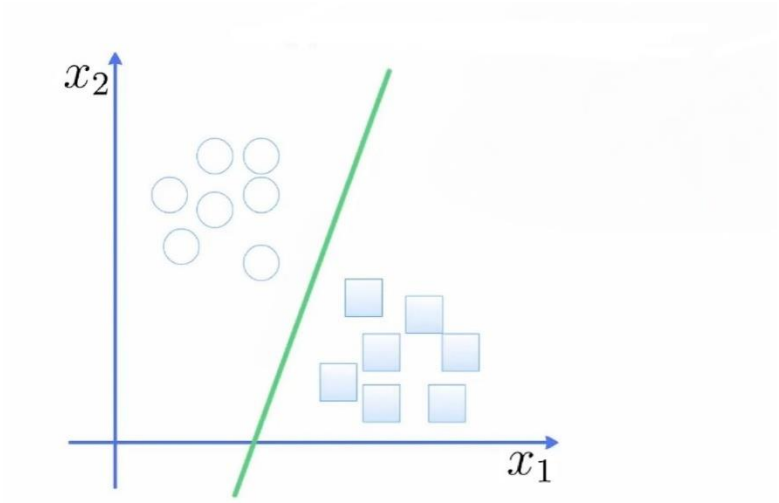
1. SVMs είναι χρήσιμες σε κείμενο και υπερκείμενο κατηγοριοποίηση , η εφαρμογή τους μπορεί να μειώσει σημαντικά την ανάγκη για την ένδειξη περιπτώσεις εκπαίδευση, τόσο στην τυπική επαγωγική και την μεταβιβαστική ρυθμίσεις. Ορισμένες μέθοδοι για ρηχή σημασιολογική ανάλυση βασίζονται σε μηχανές φορέα υποστήριξης.
2. Η ταξινόμηση των εικόνων μπορεί επίσης να πραγματοποιηθεί χρησιμοποιώντας SVM. Τα πειραματικά αποτελέσματα δείχνουν ότι τα SVM επιτυγχάνουν σημαντικά υψηλότερη ακρίβεια αναζήτησης από τα παραδοσιακά σχήματα βελτίωσης ερωτημάτων μετά από μόλις τρεις έως τέσσερις κύκλους ανατροφοδότησης σχετικότητας. Αυτό ισχύει επίσης για τα συστήματα τμηματοποίησης εικόνων, συμπεριλαμβανομένων εκείνων που χρησιμοποιούν μια τροποποιημένη έκδοση SVM που χρησιμοποιεί την προνομιακή προσέγγιση όπως προτείνεται από τον Vapnik.
3. Ταξινόμηση δορυφορικών δεδομένων όπως δεδομένα SAR χρησιμοποιώντας εποπτευόμενο SVM.
4. Χειρόγραφοι χαρακτήρες μπορούν να αναγνωριστούν χρησιμοποιώντας το SVM.
5. Ο αλγόριθμος SVM έχει εφαρμοστεί ευρέως στις βιολογικές και άλλες επιστήμες. Έχουν χρησιμοποιηθεί για την ταξινόμηση των πρωτεϊνών με έως και 90% των ενώσεων να ταξινομούνται σωστά. Έχουν προταθεί δοκιμές παραλλαγής με βάση τα βάρη SVM ως μηχανισμός ερμηνείας των μοντέλων SVM. Στο παρελθόν χρησιμοποιήθηκαν βάρη μηχανής υποστήριξης-φορέα για την ερμηνεία μοντέλων SVM. Η μεταθετική ερμηνεία των μοντέλων μηχανών υποστήριξης-φορέα για τον εντοπισμό χαρακτηριστικών που χρησιμοποιεί το μοντέλο για να κάνει προβλέψεις είναι ένας σχετικά νέος τομέας έρευνας με ιδιαίτερη σημασία στις βιολογικές επιστήμες.

3.2.2. Τρόπος Λειτουργίας

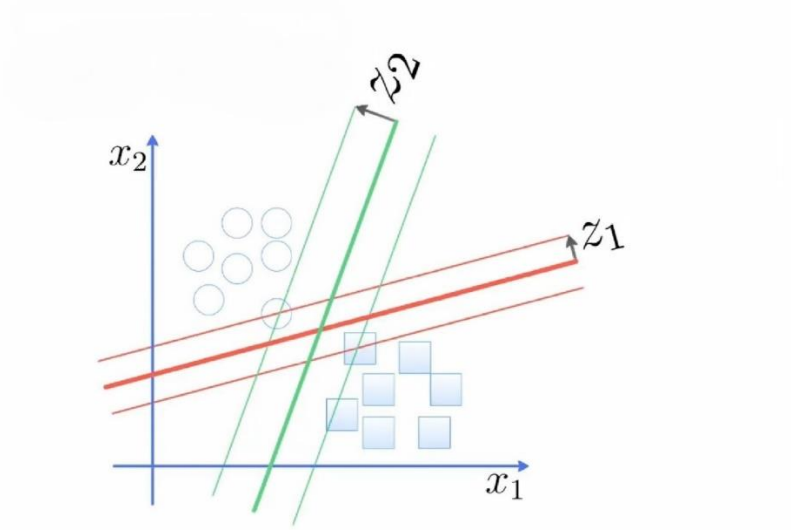
Έστω ότι έχουμε ένα καρτεσιανό επίπεδο με δύο άξονες X_1 και X_2 και πρέπει να ταξινομήσουμε όλα τα στοιχεία που βρίσκονται στο καρτεσιανό μας επίπεδο όπως φαίνονται και παρακάτω στο σχήμα.



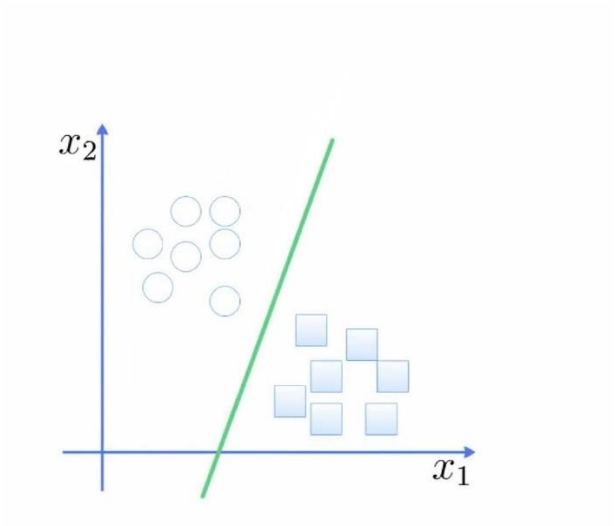
Στη συνέχεια για να ταξινομήσουμε τα στοιχεία που έχουμε πάνω στο επίπεδο μας βάζουμε ως στόχο να ταξινομηθούν όλα τα στοιχεία σε δύο κατηγορίες. Αυτό το καταφέρνουμε σχεδιάζοντας μια ευθεία γραμμή ενδιάμεσα στα κυκλικά και τα τετράγωνα στοιχεία, όπως φαίνεται και το σχήμα παρακάτω.



Βέβαια υπάρχουν διάφορες κλίσεις που μπορείς να σχεδιάσεις την ευθεία γραμμή πάνω στο καρτεσιανό επίπεδο, όπως φαίνονται και στο σχήμα παρακάτω.



Εμείς βέβαια θα πρέπει να διαλέξουμε την καλύτερη επιλογή η οποία θα έχει το μέγιστο περιθώριο από τις δύο κατηγορίες. Όπου στην δική μας περίπτωση είναι Z_2 ($Z_2 > Z_1$).

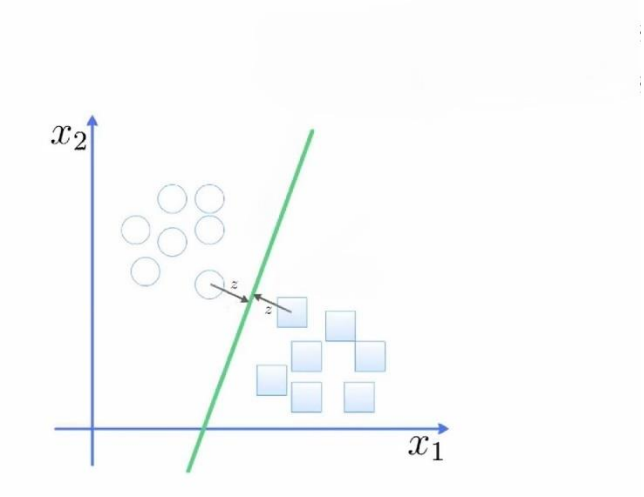


$$g(\vec{x}) \geq 1, \forall \vec{x} \in class1\ (\kappa\acute{\upsilon}\kappa\lambda o\iota)$$

$$g(\vec{x}) \leq -1, \forall \vec{x} \in class2\ (\tau\varepsilon\tau\rho\acute{\alpha}\gamma\omega\nu\alpha)$$

Έστω ότι η απόσταση (Z) μεταξύ της καλύτερης επιλογής γραμμής και του πιο κοντινού στοιχείου από κάθε κατηγορία είναι τουλάχιστον 1. Οπότε με βάση τη γεωμετρία έχουμε ότι

$$Z = \frac{|g(\vec{x})|}{||\vec{\omega}||} = \frac{1}{||\vec{\omega}||}$$



Οπότε με βάση τα παραπάνω έχουμε ότι το συνολικό περιθώριο υπολογίζεται από τον παρακάτω τύπο:

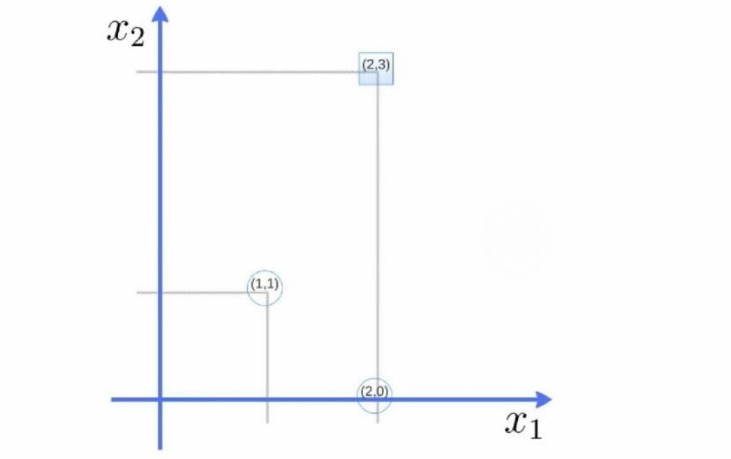
$$\frac{1}{||\vec{\omega}||} + \frac{1}{||\vec{\omega}||} = \frac{2}{||\vec{\omega}||}$$

Και εάν ελαχιστοποιήσουμε των παρονομαστή στο τέλος θα έχουμε σαν αποτέλεσμα να μεγιστοποιηθεί η διαχωριστικότητα. Για την ελαχιστοποίηση του $\vec{\omega}$ είναι μια εργασία μη γραμμικής βελτιστοποίησης η οποία μπορεί να λυθεί από τις Karush-Kuhn-Tucker (KKT) συνθήκες, χρησιμοποιώντας πολλαπλασιαστές Langrange λ_ι

$$\vec{\omega} = \sum_{\iota=0}^{N} \lambda_\iota y_i \vec{x}_i$$

$$\sum_{i=0}^{N} \lambda_i \, y_i = 0$$

Πάμε να δούμε σε ένα παράδειγμα την όλη διαδικασία. Έστω ότι έχουμε το παρακάτω καρτεσιανό επίπεδο:



Από το παραπάνω καρτεσιανό επίπεδο έχουμε ότι το

$$\vec{\omega} = (2,3) - (1,1) = (\alpha, 2\alpha)$$

και για το στοιχείο στη θέση (1,1) έχουμε ότι

g(1,1) = -1 (διότι το (1,1) είναι κύκλος)

όπως και για το στοιχείο στη θέση (2,3) έχουμε ότι

g(2,3) = 1 (διότι το (2,3) είναι τετράγωνο)

Από όλα τα παραπάνω έχουμε ότι

$$\vec{\omega} = (\alpha, 2\alpha),$$

α+2α+ω_0= -1, (από το σημείο (1,1)) και

2α+6α+ω_0= 1, (από το σημείο (2,3)

Από τις παραπάνω εξισώσεις έχουμε ότι

$$2\alpha + 6\alpha + \omega_0 = 1 \Rightarrow 8\alpha + \omega_0 = 1 \Rightarrow \omega_0 = 1 - 8\alpha$$

την παραπάνω εξίσωση την αντικαθιστούμε στην εξίσωση από το σημείο (1,1)

$$3\alpha + \omega_0 = -1 \Rightarrow 3\alpha + 1 - 8\alpha = -1 \Rightarrow 8\alpha - 3\alpha = 1 + 1 \Rightarrow 5\alpha = 2 \Rightarrow \alpha = {}^{2}/_{5}$$

Με βάση το αποτέλεσμα που βρήκαμε πάμε και λύνουμε την εξίσωση του ω_0, οπότε έχουμε

$$\omega_0 = 1 - 8\alpha \Rightarrow \omega_0 = 1 - 8 * \frac{2}{5} = \frac{5 - 16}{5} \Rightarrow \omega_0 = -\frac{11}{5}$$

Οπότε από όλα τα παραπάνω καταλήγουμε στο

$$\vec{\omega} = (\frac{2}{5}, \frac{4}{5})$$

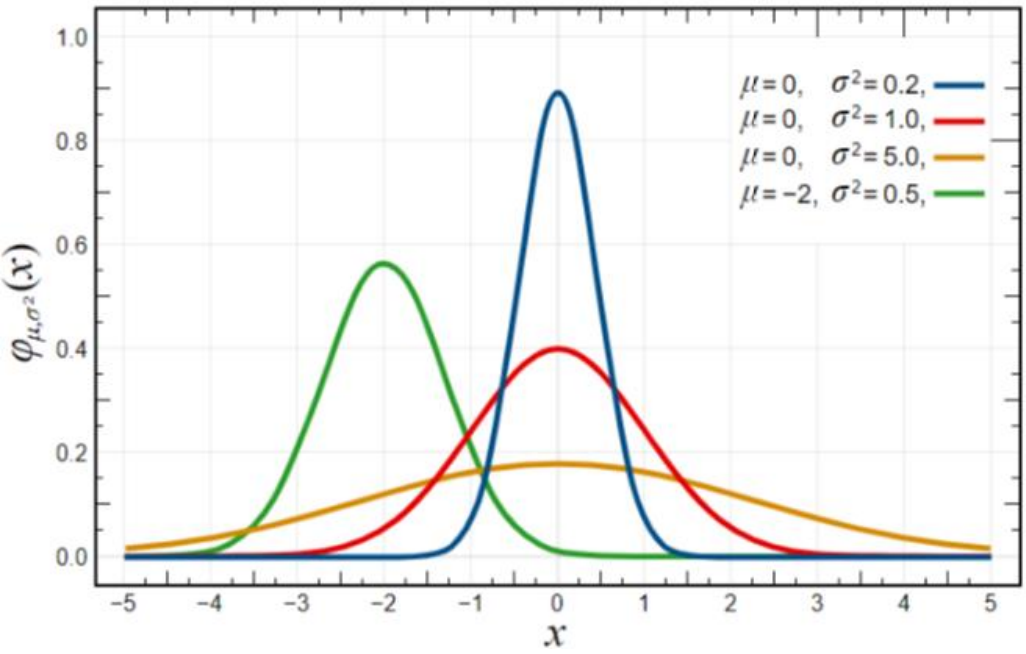
$$g(\vec{x}) = \frac{2}{5} * x_1 + \frac{4}{5} * x_2 - \frac{11}{5} \Rightarrow g(\vec{x}) = x_1 + 2x_2 - 5.5$$

Όπου το $\vec{\omega} = (\frac{2}{5}, \frac{4}{5})$ είναι αυτό που λέμε Support Vectors και η τελική μας εξίσωση $g(\vec{x}) = x_1 + 2x_2 - 5.5$ είναι αυτό που λέμε Support Vector Machine.

3.3. Gaussian Process

3.3.1. Εισαγωγή

Στην καθημερινότητα μπορεί να δημιουργηθούν ερωτήματα όπου για την απάντηση τους έχουμε αμφιβολίες και δεν μπορούμε να βρούμε μια ντετερμινιστική απάντηση, διότι γνωρίζουμε ότι για το κάθε ερώτημα υπάρχουν διάφορες περιπτώσεις όπου η απάντηση θα διαφέρει σε κάθε περίπτωση. Έτσι η καλύτερη απάντηση που μπορούμε να δώσουμε είναι ο μέσος όρος, όπου σαν όρος εμποδίζει την αβεβαιότητα. Συνήθως η αβεβαιότητα για ορισμένα θέματα μας οδηγεί στην έννοια μια τυχαίας μεταβλητής, η οποία διαμορφώνεται μαθηματικά από μια κατανομή πιθανότητας. Μια κανονική κατανομή ή Gaussian είναι μια κατηγορία διανομών που έχει σχήμα καμπάνας, όπως φαίνεται παρακάτω:



και εκφράζεται με τον παρακάτω τύπο:

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2 * \pi * \sigma^2}} * e^{-\frac{(x-\mu)^2}{2*\sigma^2}}$$

όπου:

1. μ: είναι ο μέσος όρος ή η προσδοκία της κατανομής (και επίσης η διάμεση και η λειτουργία της).
2. σ: είναι η τυπική απόκλιση.
3. σ^2: είναι η διακύμανση.

Με τον παραπάνω τύπο εκφράζεται η αβεβαιότητα για μια περίπτωση. Όταν όμως θέλουμε να εκφράσουμε την αβεβαιότητα για πολλαπλές περιπτώσεις, τότε χρησιμοποιούμε τον παρακάτω τύπο:

$$N(x|\mu,\Sigma) \stackrel{\text{def}}{=} \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]$$

Όπου ο Σ είναι ένας πίνακας συνδιακύμανσης. Τα σύμβολα λειτουργίας N και f χρησιμοποιούνται εναλλακτικά.

Εμείς για την ακρίβεια θα ασχοληθούμε με το "Gaussian" classifier. Ονομάζεται "Gaussian" classifier λόγω της παραδοχής ότι το p(x|y=c) είναι κατανομή Gauss. Είναι επίσης γνωστό ως "Mixture Gaussian" όπως και "Discriminant" classifier.

Υπάρχουν δύο παραλλαγές του ταξινομητή Gauss ανάλογα με το εάν οι πίνακες συνδιακύμανσης των τάξεων θεωρείται ότι είναι ίσοι ή όχι. Η υπόθεση μήτρας συνδιακύμανσης επηρεάζει το όριο της τάξης. Ο κοινός πίνακας συνδιακύμανσης οδηγεί στο γραμμικό όριο ενώ οι ξεχωριστοί πίνακες συνδιακύμανσης οδηγούν στο τετραγωνικό όριο.

3.3.2. Τρόπος Λειτουργίας

Για να κατανοήσουμε το πώς λειτουργεί η Gaussian Process ας φανταστούμε ότι μας δίνεται ένα σύνολο δεδομένων εκπαίδευσης που εμπίπτει σε δυο κατηγορίες (0 και 1). Ο στόχος είναι να προβλέψουμε την κατηγορία στην οποία ανήκουν τα νέα δεδομένα. Με άλλα λόγια, για τα νέα δεδομένα (x), θέλουμε να εκτιμήσουμε p(y=0|x) και p(y=1|x). Το x αντιστοιχεί σε οποιαδήποτε κλάση που έχει την υψηλότερη πιθανότητα. Με βάση το θεώρημα του Bayes μπορεί να γίνει μια προσέγγιση για την εκτίμηση του p(y=0|x) και p(y=1|x). Αυτό γίνεται με βάση τον παρακάτω τύπο

$$p(y = c|x) = \frac{p(x|y = c)p(y = c)}{\Sigma_{c'} p(x|y = c')p(y = c')}$$

Το p(y=c|x) είναι το p(y=0|x) και το p(y=1|x). Το p(y=c|x) θεωρείται ότι είναι Gaussian κατανομή. Το p(y=c) είναι μια προγενέστερη κλάση, η οποία είναι αναλογία κλάσης c/ συνολικού δείγματος. Για την υλοποίηση της ταξινόμησης ενδιαφερόμαστε μόνο για τον υποψήφιο, ώστε να μπορούμε να αγνοήσουμε τη σταθερά ομαλοποίησης που στοχεύει να κάνει την τάξη μεταγενέστερη μια έγκυρη κατανομή πιθανότητας. Όλοι οι όροι στον υποψήφιο μπορούν να εκτιμηθούν από το σύνολο δεδομένων εκπαίδευσης.

ΚΕΦΑΛΑΙΟ 4

4.1. ΚΩΔΙΚΑΣ

```
%**********Initialize variables, file paths etc
clearvars;

%Create a Structure to store the data
people = struct('name', [], 'class', [], 'type', [], 'hog', [],
'pca', [], 'lbp', []);

%Variables used to hold the total time for each algorithm
timeSVM = 0;
timeRF = 0;
timeGP = 0;

%**********Edit Images to fit our purpose
%Get the Source Images folder
pTrainingPath = uigetdir('C:\Users', 'Select Positive Training
Images Folder');
nTrainingPath = uigetdir('C:\Users', 'Select Negative Training
Images Folder');
pTestingPath = uigetdir('C:\Users', 'Select Positive Testing Images
Folder');
nTestingPath = uigetdir('C:\Users', 'Select Negative Testing Images
Folder');

%Get the Image Output folder path
outPath = uigetdir('C:\Users', 'Select output path');

%Add the proper suffixes to the paths
outPath = strcat(outPath, '\Processed Images');
pTrainingOutPath = strcat(outPath, '\PositiveTraining');
nTrainingOutPath = strcat(outPath, '\NegativeTraining');
pTestingOutPath = strcat(outPath, '\PositiveTesting');
nTestingOutPath = strcat(outPath, '\NegativeTesting');

%Create the folder structure
mkdir(sprintf('%s', outPath));
mkdir(sprintf('%s', pTrainingOutPath));
mkdir(sprintf('%s', nTrainingOutPath));
mkdir(sprintf('%s', pTestingOutPath));
mkdir(sprintf('%s', nTestingOutPath));

fixImages(pTrainingPath, pTrainingOutPath);
fixImages(nTrainingPath, nTrainingOutPath);
fixImages(pTestingPath, pTestingOutPath);
fixImages(nTestingPath, nTestingOutPath);

%**********Generate the Data
counter = 1;
%---Positive Training Initializations
imagefiles = dir(fullfile(pTrainingOutPath,'*.jpg'));
%Create the sub-Struct array for the Training Positives
```

```
for i=1:length(imagefiles)
 people(counter).name = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 people(counter).class = '1';
 people(counter).type = 'Training';
 counter = counter + 1;
end

%---Negative Training Initializations
imagefiles = dir(fullfile(nTrainingOutPath,'*.jpg'));
%Create the sub-Struct array for the Training Negatives
for i=1 : length(imagefiles)
 people(counter).name = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 people(counter).class = '0';
 people(counter).type = 'Training';
 counter = counter + 1;
end

%---Positive Testing Initializations
imagefiles = dir(fullfile(pTestingOutPath,'*.jpg'));
%Create the sub-Struct array for the Testing Positives
for i=1:length(imagefiles)
 people(counter).name = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 people(counter).class = '1';
 people(counter).type = 'Testing';
 counter = counter + 1;
end

%---Negative Testing Initializations
imagefiles = dir(fullfile(nTestingOutPath,'*.jpg'));
%Create the sub-Struct array for the Testing Negatives
for i=1 : length(imagefiles)
 people(counter).name = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 people(counter).class = '0';
 people(counter).type = 'Testing';
end

counter = 1;

%---Compile Positive Training Set
imagefiles = dir(fullfile(pTrainingOutPath,'*.jpg'));
nfiles = length(imagefiles); % Number of files found
for i=1:nfiles
 currentfilename = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 currentimage = imread(currentfilename);
 people(counter).hog = getHOG(currentimage);
 people(counter).pca = getPCA(currentimage);
 people(counter).lbp = getLBP(currentimage);
 counter = counter + 1;
end
```

```
%---Compile Negative Training Set
imagefiles = dir(fullfile(nTrainingOutPath,'*.jpg'));
nfiles = length(imagefiles);
for i=1:nfiles
 currentfilename = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 currentimage = imread(currentfilename);
 people(counter).hog = getHOG(currentimage);
 people(counter).pca = getPCA(currentimage);
 people(counter).lbp = getLBP(currentimage);
 counter = counter + 1;
end

%---Compile Positive Testing Set
imagefiles = dir(fullfile(pTestingOutPath,'*.jpg'));
nfiles = length(imagefiles); % Number of files found
for i=1:nfiles
 currentfilename = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 currentimage = imread(currentfilename);
 people(counter).hog = getHOG(currentimage);
 people(counter).pca = getPCA(currentimage);
 people(counter).lbp = getLBP(currentimage);
 counter = counter + 1;
end

%---Compile Negative Testing Set
imagefiles = dir(fullfile(nTestingOutPath,'*.jpg'));
nfiles = length(imagefiles); % Number of files found
for i=1:nfiles
 currentfilename = fullfile(imagefiles(i).folder,
imagefiles(i).name);
 currentimage = imread(currentfilename);
 people(counter).hog = getHOG(currentimage);
 people(counter).pca = getPCA(currentimage);
 people(counter).lbp = getLBP(currentimage);
 people(counter).type = 'Testing';
end
save('people.mat', 'people');
%**********Do the Machine Learning initializations
clearvars;
load people;
featuresTrain = [];
classesTrain = [];
featuresTest = [];
classesTest = [];
trainPointer = 1;
testPointer = 1;
for i=1:length(people)
 hog = people(i).hog(:);
 pca = people(i).pca(:);
 lbp = people(i).lbp(:);
 features = reshape([hog; pca; lbp], [], 1)';
```

```
 if(strcmp(people(i).type, 'Training'))
 featuresTrain(trainPointer, :) = features;
 classesTrain(trainPointer) = people(i).class;
 trainPointer = trainPointer + 1;
 else
 featuresTest(testPointer, :) = features;
 classesTest(testPointer) = people(i).class;
 testPointer = testPointer + 1;
 end
end

classesTrain = transpose(classesTrain);
classesTest = transpose(classesTest);
classesTrain = double(classesTrain);
classesTest = double(classesTest);
featuresTrain = double(featuresTrain);
featuresTest = double(featuresTest);

tic;
%Gaussian Process
trainingGaussianProcessClassifier = fitrgp(featuresTrain,
classesTrain);
resultsGP = predict(trainingGaussianProcessClassifier,
featuresTest);
timeGP = timeGP + toc;

tic;
%SVM
trainingSVMclassifier = fitcsvm(featuresTrain,
classesTrain);
resultsSVM = predict(trainingSVMclassifier,
featuresTest);
timeSVM = timeSVM + toc;

tic;
%Random Forest
trainingSVMclassifier = fitcensemble(featuresTrain,
classesTrain);
resultsRF = predict(trainingSVMclassifier, featuresTest);
timeRF = timeRF + toc;


if(timeGP > timeRF)
    if(timeRF > timeSVM)
        disp('The fastest of the 3 algorithms is Support
Vector Machine with a time of: ')
        timeSVM
    else
        disp('The fastest of the 3 algorithms is Random
Forest with a time of: ')
        timeRF
else
```

```
    disp('The fastest of the 3 algorithms is Gaussian
Process with a time of: ')
    timeGP
end

%**********Take the best answer
finalResults = zeros(length(resultsSVM));
for i = 1 : length(resultsSVM)
 num = str2double(resultsSVM(i)) + str2double(resultsRF(i)) +
str2double(resultsGP(i));
 if(num >= 2)
 num = 1;
 else
 num = 0;
 end

end

totalClasses = length(classesTest);
correctResults = 0;

for i = 1 : length(finalResults)
 if(finalResults(i) == classesTest(i))
 correctResults = correctResults + 1;
 end
end

percentage = (correctResults/totalClasses)*100.0;
message = sprintf(The results were '%0.2' persent correct.,
percentage)

function fixImages(sourcePath, outputPath)
 %Go to the image directory and get a reference for each image
 cd(sourcePath);
 imagefiles = [dir('*.jpg');dir('*.png')];
 numberOfFiles = length(imagefiles);% Number of files found
 for i = 1 : numberOfFiles
 currentfilename = imagefiles ( i ).name;
 currentimage = imread ( currentfilename );

 B = imresize(currentimage,[134 70]);

 baseFileName = sprintf('Image #%d.jpg', i);
 fullFileName = fullfile(outputPath, baseFileName);
 imwrite(B, fullFileName);
 end
end

function h = getHOG(curImg)
 h = hog_feature_vector(curImg);
end

function p = getPCA(curImg)
```

```
 pcaImage = im2double(curImg);
 RED = pcaImage(:,:,1);
 RED = RED(:);
 GREEN = pcaImage(:,:,2);
 GREEN = GREEN(:);
 BLUE = pcaImage(:,:,3);
 BLUE = BLUE(:);
 RGB = [RED,GREEN,BLUE];
 resultPCA = pca(RGB);
 p = transpose(resultPCA(:));
end

function l = getLBP(curImg)
 resultLBP = efficientLBP(curImg);
 l = transpose(resultLBP(:));
end
```

4.2. Επεξήγηση Κώδικα και Συμπεράσματα

4.2.1. Επεξήγηση Κώδικα

Στην αρχή του κώδικα μας δημιουργούμε το “clearvars” το οποίο σβήνει τις ήδη υπάρχουσες μεταβλητές από την μνήμη. Μετά δημιουργούμε ένα “struct” για να αποθηκεύσουμε τις πληροφορίες σχετικά με τις εικόνες. Έπειτα μέσω της uigetdir παίρνουμε τα μονοπάτια των φακέλων που περιέχουν τις εικόνες για το training και το testing και τον φάκελο που θα γίνει η έξοδος. Ύστερα έχουμε την δημιουργία σωστού μονοπατιού για την έξοδο των εικόνων μετά την επεξεργασία τους. Το επόμενο βήμα είναι (mkdir), το ανέκαθεν μονοπάτι μετατρέπετε σε string και χρησιμοποιείτε για να δημιουργηθεί αντίστοιχος φάκελος.

Στην συνέχεια δημιουργούμε την function fixImages η οποία παίρνει τις εικόνες από ένα φάκελο, τις μετατρέπει και τις τοποθετεί σε ένα φάκελο-έξοδος. Έπειτα απ’ όλα αυτά φτιάχνουμε έναν μετρητή για όλα και τον αρχικοποιούμε με ένα. Μετέπειτα παίρνουμε όλες τις εικόνες από τον πρώτο φάκελο και τις αποθηκεύουμε στο Imagefiles. Για κάθε εικόνα στο Imagefiles κάνουμε τα εξής: παίρνουμε το πλήρες μονοπάτι της, το οποίο αποτελείτε από το φάκελο που βρίσκετε και τον τίτλο της και το αποθηκεύουμε στην αντίστοιχοι γραμμή του people στο χαρακτηριστικό name.

Ύστερα βάζουμε στο χαρακτηριστικό class την τιμή 1 (που σημαίνει ότι υπάρχει άνθρωπος στην εικόνα, αν δεν υπήρχε θα βάζαμε 0). Κατόπιν στο χαρακτηριστικό type βάζουμε το Training (αν την χρησιμοποιούμε για Testing θα γράψουμε Testing). Μετά αυξάνουμε τον μετρητή κατά ένα. Ύστερα χρησιμοποιούμε τον μετρητή για να συνεχίσουμε από το σημείο που σταματήσαμε στον people, κάνουμε το ίδιο αντίστοιχα και για τους άλλους τρείς φακέλους εικόνων(πχ για μία εικόνα στο nTestingOutPath θα βάλουμε class 0 και type Testing). Έπειτα αρχικοποιούμε τον μετρητή στο ένα, παίρνουμε όλες τις εικόνες από το φάκελο και τις αποθηκεύουμε στο Imagefiles και μετά παίρνουμε το πλήθος των εικόνων και το βάζουμε στο nfiles

(δεν διαφέρει από τον τρόπο που το κάναμε επάνω). Για κάθε μία από αυτές τις εικόνες παίρνουμε το πλήρες μονοπάτι της (φάκελος + τίτλος αρχείου) και το χρησιμοποιούμε για να διαβάσουμε την αντίστοιχη εικόνα.

Μετέπειτα, εκτελούμε τις συναρτήσεις getHOG, getPCA, getLBP στην εικόνα και αποθηκεύουμε τα αποτελέσματα τους στην σωστή γραμμή του people, στα χαρακτηριστικά hog, pca, lbp, αντίστοιχα και αυξάνουμε τον μετρητή κατά ένα, κάνουμε το ίδιο για τους άλλους τρείς φακέλους αντίστοιχα. Τέλος, σώζουμε τον people στο αρχείο people.mat. Στη συνέχεια σβήνουμε τις μεταβλητές από την μνήμη και φορτώνουμε το αρχείο people.mat.

Στη συνέχεια αρχικοποιούμε τον πίνακα featuresTrain (εδώ θα αποθηκέψουμε τα δεδομένα για κάθε εικόνα που θα χρησιμοποιηθεί στο training κομμάτι), έπειτα αρχικοποιούμε τον πίνακα classesTrain (εδώ θα αποθηκεύσουμε της κλάσης των εικόνων που θα χρησιμοποιηθούν στο training κομμάτι). Ύστερα αρχικοποιούμε τον πίνακα featuresTest (εδώ θα αποθηκέψουμε τα δεδομένα για κάθε εικόνα που θα χρησιμοποιηθεί στο testing κομμάτι), μετέπειτα αρχικοποιούμε τον πίνακα classesTest (εδώ θα αποθηκεύσουμε τις κλάσεις των εικόνων που θα χρησιμοποιηθούν στο testing κομμάτι).

Μετά από όλα αυτά φτιάχνουμε έναν μετρητή trainPointer για τους πίνακες featuresTrain και classesTrain και έναν μετρητή testPointer για τους πίνακες featuresTest και classesTest και τους αρχικοποιούμε με την τιμή ένα. Για κάθε μία γραμμή στον people αποθηκεύουμε προσωρινά τα χαρακτηριστικά hog, pca, lbp στις μεταβλητές hog, pca, lbp αντίστοιχα. Μετατρέπουμε τους τρείς αυτούς πίνακες σε μονοδιάστατους και τους ενώνουμε κατά μήκος. Αν η εικόνα στην οποία ασχολούμαστε θα χρησιμοποιηθεί στο training κομμάτι τότε παίρνουμε την γραμμή που μόλις φτιάξαμε και την αποθηκεύουμε στον πίνακα featuresTrain στην σωστή θέση. Μετά παίρνουμε την κλάση της εικόνας και την αποθηκεύουμε στην σωστή θέση του πίνακα classesTrain και αυξάνουμε τον trainPointer κατά ένα. Αλλιώς αν χρησιμοποιηθεί για testing παίρνουμε την γραμμή που μόλις φτιάξαμε και το αποθηκεύουμε στον πίνακα featuresTest στην σωστή θέση. Μετά παίρνουμε την κλάση της εικόνας και την αποθηκεύουμε στην σωστή θέση του πίνακα classesTest και αυξάνουμε τον testPointer κατά ένα. Έπειτα αναστρέφουμε τους πίνακες classesTrain και classesTest, στη συνέχεια μετατρέπουμε σε double τα νούμερα από τους πίνακες classesTrain, featuresTrain, featuresTest και classesTest.

Ύστερα δημιουργούμε έναν trainingGaussianProcessClassifier και το εκπαιδεύουμε με τα δεδομένα που έχουμε (featuresTrain , classesTrain). Με τον ταξινομητή (classifier) που μόλις εκπαιδεύσαμε, το testing κομμάτι χρησιμοποιώντας τον featuresTest και αποθηκεύουμε τα αποτελέσματα στον resultsGP. Έπειτα κάνουμε την αντίστοιχη δουλειά για τον SVM και Random Forest.

Στη συνέχεια φτιάχνουμε έναν πίνακα ίδιου μήκους με τον αριθμό αποτελεσμάτων και τον γεμίζουμε με μηδενικά και για κάθε μία εικόνα αθροίζουμε τα αποτελέσματα των τριών ταξινομητών (SVM, Random Forest, Gaussian Process). Αν το άθροισμα είναι μεγαλύτερο ή ίσο του δύο (δηλαδή τουλάχιστον δύο από τους τρείς ταξινομητές

απάντησαν με 1) τότε βάζουμε στο finalResults το ένα, αλλιώς βάζουμε 0. Μετέπειτα αποθηκεύουμε προσωρινά το πλήθος των κλάσεων των εικόνων που χρησιμοποιήθηκαν για testing.

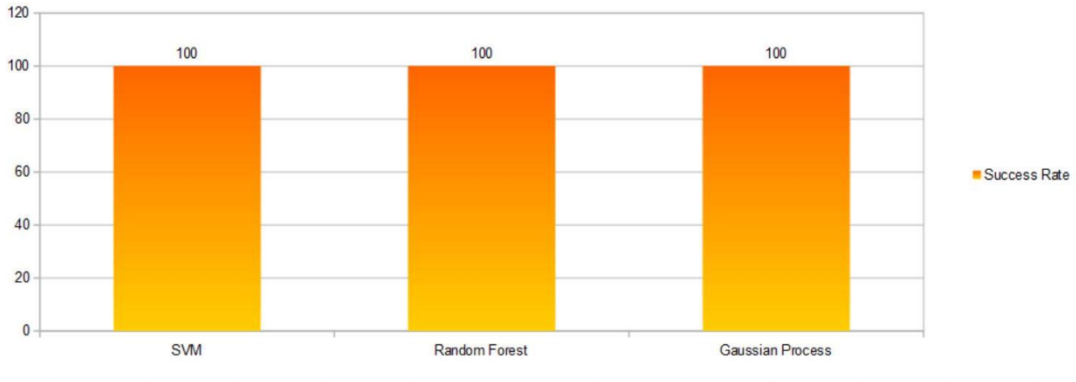
Η correctResults είναι μια μεταβλητή που θα αποθηκεύσουμε το πλήθος των αποτελεσμάτων που μαντέψαμε σωστά(μέσω των ταξινομητών). Για κάθε ένα από τα αποτελέσματα, το συγκρίνουμε με την πραγματική κλάση και αν είναι ίδια αυξάνουμε το correctResults κατά ένα. Τέλος, υπολογίζουμε το ποσοστό επιτυχίας μας και το εκτυπώνουμε.

Παρακάτω θα δούμε αναλυτικά τι κάνουν οι function που χρησιμοποιήσαμε:

1. **fixImages:** Μετακινείται στο φάκελο εισόδου. Βάζει όλες τις εικόνες στο Imagefiles. Αποθηκεύει το πλήθος αυτών στον numberOfFiles. Για κάθε μια από αυτές τις εικόνες αποθηκεύει προσωρινά το όνομα της και την διαβάζει. Αλλάζουμε το μέγεθος της εικόνας στο 134,70 και το αποθηκεύουμε στο B. Ορίζουμε το τίτλο της τελικής εικόνας, ορίζουμε το μονοπάτι της και την αποθηκεύουμε.
2. **hog:** Εκτελεί τον αλγόριθμο hog_feature_vector επάνω στην εικόνα.
3. **pca:** Μετατρέπω την εικόνα σε αριθμούς double. Παίρνω το κόκκινο κανάλι της εικόνας και το αποθηκεύω σε μια μεταβλητή με όνομα RED(κάθε εικόνα έχει τρία κανάλια red, green, blue), και παίρνω το RED και το μετατρέπω σε μονοδιάστατο πίνακα. Κάνουμε το ίδιο για το GREEN και το BLUE. Παίρνω τους μονοδιάστατους και τους συνενώνω κατά πλάτος σε ένα πίνακα RGB. Εκτελώ επάνω σε αυτόν τον πίνακα τον αλγόριθμο pca και κάνω αναστροφή στο αποτέλεσμα.
4. **lbp:** Εκτελούμε τον αλγόριθμο efficientLBP στην εικόνα και κάνουμε αναστροφή στο αποτέλεσμα.

4.2.2. Συμπεράσματα:

Ο πιο γρήγορος αλγόριθμος / ταξινομητής είναι ο SVM με 16 λεπτά, μετά ακολουθεί ο Gaussian Process με 21 λεπτά και ο πιο αργός είναι ο Random Forest με 22 λεπτά



Αφού τρέξουμε τον κώδικα μας βλέπουμε ότι λειτουργεί και ότι έχουμε 100% επιτυχία με τα συγκεκριμένα δείγματα INRIA.

4.3. Άλλες χρήσεις όμοιου / παρόμοιου Προγράμματος:

Λόγω της συνεχόμενης εξέλιξης της τεχνολογίας αυξάνονται συνεχώς και οι απαιτήσεις του ανθρώπου, έτσι με κάποιες μικρο-αλλαγές θα μπορούμε να τρέξουμε αυτό το πρόγραμμα και για άλλα ζητήματα που μας απασχολούν. Καταρχάς αυτό το πρόγραμμα αυτό θα μπορούσε να επεκταθεί πολύ εύκολα σε αναγνώριση πεζών από κινούμενη εικόνα (video). Επιπλέον θα μπορούσε να χρησιμοποιηθεί για την διαχείριση της οδικής κυκλοφορίας. Ακόμα, θα μπορούσε να χρησιμοποιηθεί σε θέματα ασφαλείας όπως η παρακολούθηση, η προστασία κλπ. Με κάποιες αλλαγές στον κώδικα θα μπορούσαμε να τον χρησιμοποιήσουμε και σε ιατρικές εικόνες για απόφαση αν ο ασθενείς έχει κάποια νόσο ή όχι, για παράδειγμα σε εικόνες με εγκεφάλους να μπορούσε να αναγνωρίζει αν ο ασθενείς έχει υποστεί εγκεφαλικό ή όχι. Τέλος, θα μπορούσε να χρησιμοποιηθεί κάτι παρόμοιο για αναγνώριση προσώπων σε βάσης δεδομένων στα αστυνομικά τμήματα.

Βιβλιογραφία

1. https://en.wikipedia.org/wiki/Random_forest
2. https://en.wikipedia.org/wiki/Principal_component_analysis
3. https://en.wikipedia.org/wiki/Support_vector_machine
4. https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients
5. https://en.wikipedia.org/wiki/LBP
6. https://eclass.uniwa.gr/modules/document/file.php/MSCIOT101/%CE%94%CE%B9%CE%B1%CE%BB%CE%AD%CE%BE%CE%B5%CE%B9%CF%82%20%CE%96%CF%8E%CE%B7%CF%82/2_PCA_EE_UNIWA.pdf
7. https://nemertes.lis.upatras.gr/jspui/bitstream/10889/8221/1/MSc_%CE%A0%CF%84%CF%85%CF%87%CE%B9%CE%B1%CE%BA%CE%B7_Final.pdf
8. https://apothesis.lib.teicrete.gr/bitstream/handle/11713/3676/StamatakisXenofon2013.pdf?sequence=1&isAllowed=y
9. http://nefeli.lib.teicrete.gr/browse/stef/epp/2012/GiakoumakisEvangelos/attached-document-1338447048-440776-19021/Giakoumakis2012.pdf
10. http://www.cs.uoi.gr/tech_reports//publications/MT-2018-5.pdf
11. https://nemertes.lis.upatras.gr/jspui/bitstream/10889/6438/1/Master_Thesis_Sotiris%20Balkouras.pdf
12. http://ikee.lib.auth.gr/record/290838/files/PTYXIAKH_ERGASIA.pdf
13. http://ikee.lib.auth.gr/record/291635/files/%CE%94%CE%B9%CE%B9%CF%80%CE%BB%CF%89%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%AE.pdf
14. http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/11ImageProc/15PCA.pdf
15. http://pascal.inrialpes.fr/data/human/
16. https://pdfs.semanticscholar.org/c21b/8b8cd5708111ea21d8095c075aef1598efcd.pdf
17. https://en.wikipedia.org/wiki/Gaussian_function
18. https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/?fbclid=IwAR3UwAA-KDY7UbNlDkM3MK95nWxr84SYmOBZcwF7oD8dQJm9uitp_vp-dSE
19. https://hal.inria.fr/inria00548512/document?fbclid=IwAR3OFVBxuFTrph7Eeos41OAdiVfEXVAtEEdtqaa3htabR6908JpDD3YTm0w
20. https://pdfs.semanticscholar.org/7cc8/3e98367721bfb908a8f703ef5379042c4bd9.pdf
21. https://www.shirin-glander.de/2018/10/ml_basics_rf/?fbclid=IwAR0HTNbbkX0XJ9IOs-19Pycc-

D6V1RbOFY-qCAI_kTW4Xl2uxlPhp2v_53U

22. http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf?fbclid=IwAR1I0CWMVHTmuJgGoJaYnOro6bGJx3eOA_nbv3-Nj-JsvUhfoZH_BTHa0cM
23. https://www.lpsm.paris/pageperso/has/source/Hand-on-ML.pdf?fbclid=IwAR3Roc1hccJ0-JSLNpWB_BHIMW6Q7_GNgLOc7lkgXHDYB6vuV8TBPUIHmo0

Παράρτημα:

Κώδικας για το LBP

```
function LBP= efficientLBP(inImg, varargin) % filtR, isRotInv, isChanWiseRot
%% efficientLBP
% The function implements LBP (Local Binary Pattern analysis).
%
%% Syntax
% LBP= efficientLBP(inImg);
%
%% Description
% The LBP tests the relation between pixel and it's neighbors, encoding this relation into
% a binary word. This allows detection of patterns/features.
% The function is inpired by materials published by Matti Pietik?inen in
% http://www.cse.oulu.fi/CMV/Research/LBP . This implementation hovewer is not totally
% allighned with the mthods proposed by Professor Pietik?inen (see Issues & Comments).
%
%% Input arguments (defaults exist):
% inImg- input image, a 2D matrix (3D color images will be converted to 2D intensity
% value images)
% filtR- a 2D matrix representing a round/radial filter. It can be generated using
% generateRadialFilterLBP function.
% isRotInv- a logical flag. When enabled generated rotation invariant LBP accuired via
% fining an angle at whihc the LBP og a given pixelis minimal. Icreases run time, and
% results in a relatively sparce hsitogram (as many combinations disappear).
% isChanWiseRot- a logical flag, when enabled (default value) allowes channel wise
% rotation. When disabled/false rotation carried out based on roation of first color
% channel. Supported only when "isEfficent" is enabled. When "isEfficent" is
% disabled "isChanWiseRot" is true.
%
%% Output arguments
% LBP- LBP image UINT8/UINT16/UINT32/UINT64/DOUBLE of same
```

```
dimentions
% [Height x Width] as inImg.
%
%% Issues & Comments
% - Currenlty, all neigbours are treated alike.
Basically, we can use wighted/shaped
% filter.
%  -  The  rotation  invariant  LBP  histogram
includes less then bins then regular LBP BY
% DEFINITION the zero trailing binary words are
excluded for example, so it can be
%   reduced   to   a   mush   more   component
representation. Actually for 8 niegbours it's
37
% bins, instead of 256. An efficnet way to
calculate those bins value is needed.
%
%% Example
% img=imread('peppers.png');
% filtR=generateRadialFilterLBP(8, 1);
% tic;
%  %  note  this  filter  dimentions  aren't
legete...
%  effLBP=  efficientLBP(img,  'filtR',  filtR,
'isRotInv', true, 'isChanWiseRot', false);
% effTime=toc;
%
% % verify pixel wise implementation returns
same results
% tic;
% % same parameters as before
%   pwLBP=pixelwiseLBP(img,   'filtR',   filtR,
'isRotInv', true, 'isChanWiseRot', false);
% inEffTime=toc;
% fprintf('\nRun time ratio %.2f. Same result
eqaulity chesk: %o.\n', inEffTime/effTime,...
% isequal(effLBP, pwLBP));
%
% figure;
% subplot(1, 3, 1)
% imshow(img);
% title('Original image');
%
% subplot(1, 3, 2)
% imshow( effLBP );
% title('Efficeint LBP image');
%
% subplot(1, 3, 3)
% imshow( pwLBP );
% title('Pixel-wise LBP image');
%
%% See also
%   pixelwiseLBP   %   a   straigh   forward
iplmenetation  of  LBP,  should  achive  same
```

```
results
% generateRadialFilterLBP % custom function
generating circulat filters
%
%% Revision history
% First version: Nikolay S. 2012-05-01.
% Last update: Nikolay S. 2014-01-09.
%
% *List of Changes:*
% 2014-01-16- support new radial filetr
generation function
% 'generateRadialFilterLBP'. The new filter is
3D shapes, and it is alighned with
% "Gray Scale and Rotation Invariant Texture
Classification with Local Binary Patterns"
%                                          from
http://www.ee.oulu.fi/mvg/files/pdf/pdf_6.pdf.
% Changed filter direction (to CCW), starting
point (3 o'clock instead of 12), support
% pixels interpolation.
% 2014-01-09- split pixel-wise implementation
to a stand-alone function. Use
% round/circular filter generated via
generateRadialFilterLBP
% 2014-01-06 isChanWiseRot flag added to allow
dictation of uniform rotation of all color
% channels. Added witbar for the disabled
'isEfficent case, so the user will see it's
% working...
% 2013-12-30 isRotInv flag added to allow
minimal avalible LBP to result in LBP that is
% rotation invaraint. When enabled, minimal
possibel LBP will be calculated, via
% rotating the neigborhood.
% Inputs style cgange- support regulat values
input, 'names' values pairs input, and
% structure (where structure filed is the
variable name, and it's conents is the value)
% input.
% 2012-08-28 Neighbours were scanned column
wise (regular Matlab way), while they should
% be scanned clock-wise/counter clock-wise
direction. A Helix/Snail indexing function
% was written and added, to deal with this
issue.
% 2012-08-27 Chris Forne comment mentioned some
erros found in the code. As I haven't
% made any use fo the code, for the last few
month, I haven't noticed the mentioned
% issues, so many thanks goes to Chris for his
sharp eye. Bugs fixes, and some
% modification intorduced.
% 2012-05-01 After writing down the primitove
version, a filtering based miplementation
```

```
% was proposed, improving run time by factor of
80-150..
%% Deafult params
isRotInv=false;
isChanWiseRot=false;
filtR=generateRadialFilterLBP(8, 1);
%% Get user inputs overriding default values
funcParamsNames={'filtR',           'isRotInv',
'isChanWiseRot'};
assignUserInputs(funcParamsNames, varargin{:});
if ischar(inImg) && exist(inImg, 'file')==2 %
In case of file name input- read graphical file
 inImg=imread(inImg);
End
nClrChans=size(inImg, 3);
inImgType=class(inImg);
calcClass='single';
isCalcClassInput=strcmpi(inImgType, calcClass);
if ~isCalcClassInput
 inImg=cast(inImg, calcClass);
End
imgSize=size(inImg);
nNeigh=size(filtR, 3);
if nNeigh<=8
 outClass='uint8';
elseif nNeigh>8 && nNeigh<=16
 outClass='uint16';
elseif nNeigh>16 && nNeigh<=32
 outClass='uint32';
elseif nNeigh>32 && nNeigh<=64
 outClass='uint64';
Else
 outClass=calcClass;
End
if isRotInv
 nRotLBP=nNeigh;
 nPixelsSingleChan=imgSize(1)*imgSize(2);
 iSingleChan=reshape(      1:nPixelsSingleChan,
imgSize(1), imgSize(2) );
Else
 nRotLBP=1;
End
nEps=-3;
weigthVec=reshape(2.^(  (1:nNeigh)  -1),  1,  1,
nNeigh);
weigthMat=repmat( weigthVec, imgSize([1, 2]) );
binaryWord=zeros(imgSize(1),        imgSize(2),
nNeigh, calcClass);
LBP=zeros(imgSize, outClass);
possibleLBP=zeros(imgSize(1),       imgSize(2),
nRotLBP);
for iChan=1:nClrChans
 %  Initiate  neighbours  relation  filter  and
LBP's matrix
```

```
 for iFiltElem=1:nNeigh
 % Rotate filter- to compare center to next
neigbour
 filtNeight=filtR(:, :, iFiltElem);

 % calculate relevant LBP elements via
filtering
 binaryWord(:, :, iFiltElem)=cast( ...
 roundnS(filter2(   filtNeight,   inImg(:,   :,
iChan), 'same' ), nEps) >= 0,...
 calcClass );
 % Without rounding sometimes inaqulity happens
in some pixels
 % compared to pixelwiseLBP
 end % for iFiltElem=1:nNeigh
 for iRot=1:nRotLBP
 % find all relevant LBP candidates
 possibleLBP(:,                                  :,
iRot)=sum(binaryWord.*weigthMat, 3);
 if iRot < nRotLBP
 binaryWord=circshift(binaryWord, [0, 0, 1]); %
shift binaryWord elements
 End
 End

 if isRotInv
 if iChan==1 || isChanWiseRot
 % Find minimal LBP, and the rotation applied
to first color channel
 [minColroInvLBP,   iMin]=min(possibleLBP,   [],
3);

 % calculte 3D matrix index
 iCircShiftMinLBP=iSingleChan+(iMin-
1)*nPixelsSingleChan;
 else
 % the above rotation of the first channel,
holds to rest of the channels
 minColroInvLBP=possibleLBP(iCircShiftMinLBP);
 end % if iChan==1 || isChanWiseRot
 Else
 minColroInvLBP=possibleLBP;
 end % if isRotInv

 if strcmpi(outClass, calcClass)
 LBP(:, :, iChan)=minColroInvLBP;
 Else
 LBP(:,      :,      iChan)=cast(minColroInvLBP,
outClass);
 End
end % for iChan=1:nClrChans
```

Κώδικας για το HOG:

```
function
[feature]        =
hog_feature_vecto
r(im)
                    % The given code finds the HOG feature vector
                    for any given image. HOG
                    % feature vector/descriptor can then be used for
                    detection of any
                    % particular object. The Matlab code provides
                    the exact implementation of
                    % the formation of HOG feature vector as
                    detailed in the paper "Pedestrian
                    % detection using HOG" by Dalal and Triggs
                    % INPUT => im (input image)
                    % OUTPUT => HOG feature vector for that
                    particular image
                    % Example: Running the code
                    % >>> im = imread('cameraman.tif');
                    % >>> hog = hog_feature_vector (im);
                    % Convert RGB iamge to grayscale
                    if size(im,3)==3
                     im=rgb2gray(im);
                    end
                    im=double(im);
                    rows=size(im,1);
                    cols=size(im,2);
                    Ix=im; %Basic Matrix assignment
                    Iy=im; %Basic Matrix assignment
                    % Gradients in X and Y direction. Iy is the
                    gradient in X direction and Iy
                    % is the gradient in Y direction
                    for i=1:rows-2
                     Iy(i,:)=(im(i,:)-im(i+2,:));
                    end
                    for i=1:cols-2
                     Ix(:,i)=(im(:,i)-im(:,i+2));
                    end
                    gauss=fspecial('gaussian',8); %% Initialized a
                    gaussian filter with sigma=0.5 * block width.
                    angle=atand(Ix./Iy); % Matrix containing the
                    angles of each edge gradient
                    angle=imadd(angle,90); %Angles in range (0,180)
                    magnitude=sqrt(Ix.^2 + Iy.^2);
                    % figure,imshow(uint8(angle));
                    % figure,imshow(uint8(magnitude));
                    % Remove redundant pixels in an image.
                    angle(isnan(angle))=0;
                    magnitude(isnan(magnitude))=0;
                    feature=[]; %initialized the feature vector
                    % Iterations for Blocks
                    for i = 0: rows/8 - 2
                     for j= 0: cols/8 -2
                     %disp([i,j])
```

```
 mag_patch = magnitude(8*i+1 : 8*i+16 , 8*j+1 :
8*j+16);
 %mag_patch = imfilter(mag_patch,gauss);
 ang_patch = angle(8*i+1 : 8*i+16 , 8*j+1 :
8*j+16);

 block_feature=[];

 %Iterations for cells in a block
 for x= 0:1
 for y= 0:1
 angleA =ang_patch(8*x+1:8*x+8, 8*y+1:8*y+8);
 magA =mag_patch(8*x+1:8*x+8, 8*y+1:8*y+8);
 histr =zeros(1,9);

 %Iterations for pixels in one cell
 for p=1:8
 for q=1:8
%
 alpha= angleA(p,q);

 % Binning Process (Bi-Linear Interpolation)
 if alpha>10 && alpha<=30
 histr(1)=histr(1)+ magA(p,q)*(30-alpha)/20;
 histr(2)=histr(2)+ magA(p,q)*(alpha-10)/20;
 elseif alpha>30 && alpha<=50
 histr(2)=histr(2)+ magA(p,q)*(50-alpha)/20;
 histr(3)=histr(3)+ magA(p,q)*(alpha-30)/20;

 elseif alpha>50 && alpha<=70
 histr(3)=histr(3)+ magA(p,q)*(70-alpha)/20;
 histr(4)=histr(4)+ magA(p,q)*(alpha-50)/20;
 elseif alpha>70 && alpha<=90
 histr(4)=histr(4)+ magA(p,q)*(90-alpha)/20;
 histr(5)=histr(5)+ magA(p,q)*(alpha-70)/20;
 elseif alpha>90 && alpha<=110
 histr(5)=histr(5)+ magA(p,q)*(110-alpha)/20;
 histr(6)=histr(6)+ magA(p,q)*(alpha-90)/20;
 elseif alpha>110 && alpha<=130
 histr(6)=histr(6)+ magA(p,q)*(130-alpha)/20;
 histr(7)=histr(7)+ magA(p,q)*(alpha-110)/20;
 elseif alpha>130 && alpha<=150
 histr(7)=histr(7)+ magA(p,q)*(150-alpha)/20;
 histr(8)=histr(8)+ magA(p,q)*(alpha-130)/20;
 elseif alpha>150 && alpha<=170
 histr(8)=histr(8)+ magA(p,q)*(170-alpha)/20;
 histr(9)=histr(9)+ magA(p,q)*(alpha-150)/20;
 elseif alpha>=0 && alpha<=10
 histr(1)=histr(1)+ magA(p,q)*(alpha+10)/20;
 histr(9)=histr(9)+ magA(p,q)*(10-alpha)/20;
 elseif alpha>170 && alpha<=180
 histr(9)=histr(9)+ magA(p,q)*(190-alpha)/20;
 histr(1)=histr(1)+ magA(p,q)*(alpha-170)/20;
 end
```

```
 end
 end
 block_feature=[block_feature      histr];      % Concatenation of Four histograms to form one block feature

 end
 end
 % Normalize the values in the block using L1-Norm

block_feature=block_feature/sqrt(norm(block_feature)^2+.01);

 feature=[feature    block_feature];    %Features concatenation
 end
end
feature(isnan(feature))=0;  %Removing Infinitiy values
% Normalization of the feature vector using L2-Norm
feature=feature/sqrt(norm(feature)^2+.001);
for z=1:length(feature)
 if feature(z)>0.2
 feature(z)=0.2;
 end
end
feature=feature/sqrt(norm(feature)^2+.001);
% toc;
```