Συσταδοποίηση ρευμάτων δεδομένων με τον αλγόριθμο D-Stream

Νίκος Γαλανάκης, Π12024

Πανεπιστήμιο Πειραιά
Τμήμα Πληροφορικής
Πτυχιακή Εργασία
Οκτώβρης, 2017

Στους γονείς μου – Ελένη & Δημήτρης

Εισαγωγικό Σημείωμα

Η παρούσα πτυχιακή εργασία προετοιμάστηκε στο Τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς, για την απόκτηση του πτυχίου Πληροφορικής. Υλοποιήθηκε κατά την περίοδο μεταξύ Σεπτέμβρη 2016 και Οκτώβρη 2017, υπό την επίβλεψη των καθηγητών κ. Γιάννη Θεοδωρίδη και κ. Νίκου Πελέκη.

Πειραιάς, Οκτώβρης 2017 *Νίκος Γαλανάκης*

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους καθηγητές μου, κ. Γιάννη Θεοδωρίδη και κ. Νίκο Πελέκη, για την πολύτιμη καθοδήγησή τους καθ' όλη τη διάρκεια της συγγραφής της πτυχιακής μου εργασίας. Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την υποστήριξη και την κατανόησή τους.

Περίληψη

Στην παρούσα πτυχιακή εργασία επιχειρείται η εφαρμογή clustering σε data stream, χρησιμοποιώντας μια density-based προσέγγιση, με υλοποίηση του αλγορίθμου D-Stream. Ο αλγόριθμος χωρίζεται σε δύο στάδια: το online και το offline. Το online στάδιο, αφενός, αναθέτει τα εισερχόμενα από το stream δεδομένα στα grids στα οποία αντιστοιχούν, αφετέρου το offline στάδιο υπολογίζει το grid density και δημιουργεί clusters από τα ενεργά grids βάσει του density που έχουν. Υιοθετείται παράλληλα μια τεχνική για τη μείωση του density των grids με βάση το χρόνο, ώστε να αποτυπώνονται δυναμικά οι αλλαγές που προκύπτουν στα δεδομένα από το stream. Επιπροσθέτως, παρουσιάζεται μια τεχνική για τον εντοπισμό και την αφαίρεση των sporadic grids, τα οποία δημιουργούνται από το θόρυβο που έχει το data stream, ώστε να μην καταναλώνεται χώρος από το σύστημα και να μην επηρεάζεται η αποδοτικότητα του αλγορίθμου. Ο αλγόριθμος D-Stream κάνει την εφαρμογή clustering σε high-speed streaming εφικτή, χωρίς να επηρεάζεται η ποιότητα των clusters.

Περιεχόμενα

1

Εισαγωγή

Με την εξέλιξη της τεχνολογίας είναι εφικτή η δυνατότητα να συλλέγεται μεγάλος όγκος από πραγματικά δεδομένα σε πραγματικό χρόνο. Οικονομικές συναλλαγές, αναλύσεις από ιστότοπους, παρακολούθηση καιρού από αισθητήρες, GPS δεδομένα, TCP/IP κίνηση σε δίκτυα, είναι μερικά μόνο παραδείγματα από data streams. Η σχεδίαση και η εφαρμογή γρήγορων, αποτελεσματικών και ποιοτικά σωστών αλγορίθμων για την αξιοποίηση όλων αυτών των δεδομένων, όπως για παράδειγμα το να παρθούν αποφάσεις σε πραγματικό χρόνο για συγκεκριμένα προβλήματα, αποτελεί μια καινούργια πρόκληση.

Για να εξορυχθεί πληροφορία από μεγάλο όγκο δεδομένων, έχουν υλοποιηθεί data mining τεχνικές, όπως clustering και classification. Όμως, οι συμβατικές data mining μέθοδοι που χρησιμοποιούνται για στατικά datasets δεν είναι κατάλληλες για data streaming mining. Αυτό οφείλεται στο γεγονός πως στα data streams υπάρχει συνεχόμενη ροή δεδομένων, θεωρητικά συνεχιζόμενη επ' άπειρον, σε αντίθεση με τα στατικά datasets, όπου το σύνολο των δεδομένων μπορούμε να το γνωρίζουμε πριν από την επεξεργασία του. Αποτέλεσμα αυτής της ιδιότητας είναι η μη επιτρεπτή αποθήκευσή τους στη μνήμη ενός συστήματος, οπότε δημιουργείται η ανάγκη για γρήγορη επεξεργασία των δεδομένων τη στιγμή που εισέρχονται από το stream. Επιπλέον, εξαιτίας της συνέχειας του stream, η τυχαία επιλογή δεδομένων για επεξεργασία είναι αδύνατη, με συνέπεια τα δεδομένα να γίνονται αντικείμενο επεξεργασίας μόνο μια φορά, κατά την εισαγωγή τους. Επιπροσθέτως, πρέπει να ληφθεί υπ' όψιν ότι τα δεδομένα δημιουργούνται σε πραγματικό περιβάλλον, με αποτέλεσμα να μην ακολουθούν κάποιο σταθερό μοτίβο εξέλιξης, ώστε η τεχνική που πρέπει να υλοποιηθεί για αυτά πρέπει δυναμικά να προσαρμόζεται σε αυτήν την ιδιαιτερότητά τους. Επίσης, οι αλγόριθμοι που επεξεργάζονται τα δεδομένα πρέπει να υπολογίζουν τη βαρύτητα που έχει ο χρόνος στα δεδομένα, ώστε να μην εξάγουν αποτελέσματα με παρελθοντικά δεδομένα. Τέλος, είναι επιτακτική ανάγκη να αντιμετωπίζεται και να διαγράφεται ο θόρυβος που μπορεί να δημιουργηθεί κατά την συλλογή τους, λόγω προβληματικών αισθητήρων ή συναφών προβλημάτων.

Ο πρώτος που προσπάθησε να εφαρμόσει τεχνικές clustering σε data streams ήταν ο Aggarwal [1]. Πρότεινε έναν αλγόριθμο για clustering σε data

stream, το οποίο διαχωρίζεται αφενός στο στάδιο της online micro-clustering επεξεργασίας, κατά το οποίο από τα δεδομένα που εισέρχονται από το stream θα κρατιέται μόνο μια στατιστική περίληψή τους και αφετέρου στο στάδιο της offline macro-clusters επεξεργασίας, το οποίο θα αξιοποιεί τα αποθηκευμένα δεδομένα για να δημιουργήσει τα clusters. Ένα από τα βασικά μειονεκτήματά του είναι πως δεν μπορεί να δημιουργήσει clusters με τυχαία σχήματα, ενώ προϋποθέτει επιπλέον πως ο αριθμός των clusters που θα δημιουργηθούν πρέπει να είναι γνωστός πριν την διαδικασία του clustering.

Το density-based clustering έχει προταθεί ως άλλος σημαντικός τομέας για clustering αλγόριθμους [2]. Η density-based μέθοδος είναι κατάλληλη για βάση στους αλγόριθμους που εφαρμόζουν clustering σε data streams, επειδή έχουν τη δυνατότητα να δημιουργούν clusters με τυχαία σχήματα, αντιμετωπίζουν αποτελεσματικά το θόρυβο και χρειάζεται να επεξεργαστούν τα raw data μόνο μια φορά. Επιπλέον, σε αντίθεση με τους άλλους αλγόριθμους που έχουν υλοποιηθεί για online clustering, δεν είναι ανάγκη να γνωρίζουν τον αριθμό των clusters, καθώς τα δημιουργούν με βάση τα δεδομένα και τους γείτονές τους.

Στην παρούσα πτυχιακή εργασία επιχειρείται υλοποίηση του D-Stream αλγόριθμου, ενός density-based clustering αλγόριθμου για data streams.

Συγκεκριμένα, τα δεδομένα δεν αντιμετωπίζονται ως μια μεγάλη ακολουθία από δεδομένα, αντίθετα δίνεται βαρύτητα σε αυτά με βάση το χρόνο που εισήλθαν από το data stream. Το density τους μειώνεται εκθετικά σύμφωνα με ένα decay factor, το οποίο δηλώνει το ρυθμό που επηρεάζονται τα δεδομένα από το χρόνο και ορίζεται από τον χρήστη. Με το decay factor επιτυγχάνεται δυναμικά και αυτόματα η προσαρμογή των clusters στην εξέλιξη των δεδομένων μέσα στο χρόνο, τοποθετώντας βαρύτητα στα πιο πρόσφατα δεδομένα, χωρίς να διαγράφει τελείως τα παρελθοντικά. Επιπλέον, επειδή βασίζεται στο density-based, δε χρειάζεται ο χρήστης να ορίσει από πριν τον αριθμό των clusters.

Εξαιτίας του μεγάλου όγκου που αναφέρθηκε νωρίτερα ότι δημιουργούν τα data streams, η αποθήκευση του density για κάθε δεδομένο ξεχωριστά είναι αδύνατη. Για την αντιμετώπιση αυτού του ζητήματος, ο χώρος των δεδομένων διαιρείται σε καλά ορισμένα grids. Τα δεδομένα αντιστοιχούν σε grids και αποθηκεύεται μόνο το density των grids, ανάλογα με τον αριθμό των δεδομένων που έχουν αντιστοιχηθεί σε αυτά. Οι υπολογισμοί που γίνονται για το clustering εφαρμόζονται μόνο στα grids, με αποτέλεσμα να μην χρειάζεται να αποθηκεύονται τα raw data στο σύστημα.

Όσον αφορά σε πολυδιάστατα δεδομένα, για αυτά μπορεί να δημιουργηθεί μεγάλος αριθμός grids, για τα περισσότερα εκ των οποίων αποδεικνύεται στην πράξη πως είναι είτε κενά, είτε πως έχουν ελάχιστα δεδομένα. Αυτά αντιμετωπίζονται ως θόρυβος και διαγράφονται αποτελεσματικά.

Τα επόμενα κεφάλαια είναι οργανωμένα ως εξής: στο κεφάλαιο 2 αναλύεται το θεωρητικό υπόβαθρο για online clustering και ήδη υπάρχουσες εργασίες πάνω στο online clustering. Έπειτα, στο κεφάλαιο 3 παρατίθεται η έννοια του density grid, βασικοί ορισμοί του αλγόριθμου και η διαδικασία του clustering με βάση το density. Στο κεφάλαιο 4 γίνεται ανάλυση του D-Stream. Στη συνέχεια, στο

κεφάλαιο 5 παρουσιάζεται η υλοποίηση του D-Stream. Τέλος, στο κεφάλαιο 6 γίνεται η πειραματική αξιολόγηση του αλγόριθμου.

2

Σχετικές Εργασίες

Οι προκλήσεις προς επίλυση των online clustering algorithms χωρίζονται σε δύο κατηγορίες. Η πρώτη αφορά προβλήματα που εμφανίζονται στον τομέα της συσταδοποίησης (clustering) και η δεύτερη στην επεξεργασία των ροών δεδομένων (data streams).

Συγκεκριμένα, τα ζητήματα που απαιτούν επίλυση σχετικά με το online clustering, εντοπίζονται ως εξής: Τα δεδομένα σε μια ροή δεδομένων δημιουργούνται όσο περνάει ο χρόνος, επομένως οι συστάδες θα πρέπει να ανανεώνονται συνεχόμενα, ώστε να προσαρμόζονται στα καινούρια δεδομένα. Επιπλέον, οι αλγόριθμοι θα πρέπει να υπολογίζουν δυναμικά τον αριθμό των συστάδων με βάση τα δεδομένα που λαμβάνουν και να είναι ανεκτικοί στα outliers. Στη ροή δεδομένων τα δεδομένα δεν μπορούν να αποθηκευτούν, με αποτέλεσμα η επεξεργασία τους να γίνεται μόνο μια φορά με γρήγορη επεξεργασία από τον αλγόριθμο, ώστε να προσαρμόζεται στο ρυθμό που λαμβάνει τα δεδομένα από τη ροή δεδομένων. Τέλος, πρέπει να αντιμετωπιστεί και το πρόβλημα της περιορισμένης μνήμης.

2.1 Window Models

Για να υπολογιστούν τα δεδομένα που συμβάλλουν στους αλγορίθμους από τις ροές δεδομένων χρησιμοποιούνται window models, με τα πιο διαδεδομένα να είναι τα ακόλουθα:

Landmark Window Model: Το clustering εφαρμόζεται από ένα συγκεκριμένο σημείο στο χρόνο, το οποίο ονομάζεται landmark, μέχρι την παρούσα χρονική στιγμή. Το landmark μπορεί να ορίζεται είτε με χρονικούς όρους (εβδομαδιαία, μηνιαία), είτε με αριθμό δεδομένων (ανά 1000 δεδομένα). Όταν ξεπερνιέται το landmark, όλα τα δεδομένα που είχε το προηγούμενο διαγράφονται και ελέγχονται μόνο αυτά που είναι στο τρέχον window. Μειονεκτήματα του συγκεκριμένου window model εντοπίζονται αφενός στην ανάγκη να οριστεί ένα σωστό

landmark για την ανάλυση των δεδομένων, αφετέρου στο γεγονός ότι όλα τα δεδομένα που ανήκουν στο window αντιμετωπίζονται με την ίδια βαρύτητα.

Sliding Window Model: Το clustering γίνεται για len χρόνο πίσω από την τωρινή χρονική στιγμή t. Όσο περνάει ο χρόνος, το window έχει το ίδιο μέγεθος και διατηρεί τα δεδομένα που έχουν εισαχθεί στο χρονικό διάστημα $|t - len + 1, t|$ ενώ όσα δεδομένα είχαν μπει πριν από αυτό το διάστημα σταματούν να υπολογίζονται στο clustering. Είναι κατάλληλο για εφαρμογές που τις ενδιαφέρουν μόνο τα πιο πρόσφατα δεδομένα, αφού τα δεδομένα που είναι μέσα στο window έχουν την ίδια βαρύτητα.

Damped Window Model: Τα δεδομένα που εισέρχονται στο window έχουν βαρύτητα ανάλογη με το χρόνο που μπήκαν στο window, δηλαδή όταν μπαίνουν στο window έχουν τη μεγαλύτερη βαρύτητα, ενώ όσο περνάει ο χρόνος αυτή μειώνεται. Ο τύπος είναι

$$f = 2^{-\lambda(t_c - t_o)}$$

όπου λ είναι ο συντελεστής για τη μείωση του βάρους, t_c ο παρών χρόνος και t_o ο χρόνος εισαγωγής του στο window. Σε αντίθεση με τα προηγούμενα μοντέλα, δε διαγράφει τα παρελθοντικά δεδομένα, αλλά μειώνει την επίδρασή τους στο clustering.

2.2 Μοντέλα επεξεργασίας δεδομένων

Για την επεξεργασία δεδομένων με σκοπό το clustering ακολουθείται ένας από τους παρακάτω τρόπους:

Online Clustering: Για την εφαρμογή του clustering γίνεται ένα μόνο πέρασμα στα δεδομένα που εισέρχονται από το stream και διατηρείται μια γενική προσέγγιση, ώστε να διατηρηθούν τα clusters. Καθ' όλη τη διάρκεια του stream χρησιμοποιείται ένα μοντέλο για το clustering και στην πορεία του χρόνου γίνεται ανανέωση και προσαρμογή των ήδη υπαρχόντων clusters στα καινούργια δεδομένα που εισέρχονται από το stream. Με αυτή την προσέγγιση χάνεται η δυνατότητα να γίνει μελέτη των clusters σε διαφορετικές χρονικές στιγμές στο παρελθόν.

Online-offline clustering: Για την αντιμετώπιση του μειονεκτήματος της πρώτης μεθόδου, ο Aggarwal [1]. πρότεινε την online-offline clustering προσέγγιση. Αυτή αποτελείται από δυο στάδια clustering: το online στάδιο διατηρεί στατιστικά για τα δεδομένα που εισάγονται από το stream με τον ίδιο τρόπο που ακολουθεί και η πρώτη μέθοδος, ενώ το offline στάδιο εκτελεί το clustering πάνω στα στατιστικά που έχει από το online στάδιο σε χρονικές στιγμές που επιλέγει ο χρήστης. Με αυτόν τον τρόπο δίνεται η δυνατότητα να υπάρχει "εικόνα" για την κατάσταση των clusters σε διαφορετικές χρονικές στιγμές.

2.3 Stream Clustering

Οι διαφορετικές προσεγγίσεις για stream clustering αλγορίθμους μπορούν να κατηγοριοποιηθούν σε:

Partitioning αλγορίθμους, οι οποίοι ομαδοποιούν τα δεδομένα σε k clusters, με το k να ορίζεται από το χρήστη. Η ομαδοποίηση βασίζεται στη βελτιστοποίηση κάποιων κριτηρίων, με το πιο διαδεδομένο να είναι το Sum of Square Erros. Τα προβλήματα που παρουσιάζουν σε data streaming είναι ότι ο χρήστης πρέπει να ξέρει από την αρχή πόσα clusters θα δημιουργηθούν κατά το πέρασμα του χρόνου, ότι δημιουργούν σφαιρικά σχήματα για clusters, ενώ τα δεδομένα κατά κύριο λόγω δημιουργούν clusters που σχηματίζουν τυχαία σχήματα και ότι τέλος, δεν μπορούν να αντιμετωπίσουν το θόρυβο, που σε πραγματικά δεδομένα και σε πραγματικό χρόνο είναι δεδομένο πως θα προκύψει. Partitioning αλγόριθμοι είναι ο STREAM [2], CluStream [1], SWClustering [3] και ο StreamKM++[4]. Αρχικά, ο αλγόριθμος STREAM [2] βασίζεται στο k-median clustering και για την προσαρμογή του στο streaming χωρίζει το stream σε batches μεγέθους m. Για κάθε batch εφαρμόζεται ο αλγόριθμος k-median ώστε να βελτιστοποιήσει το sum of squared error. Ελέγχεται παράλληλα αν τα medians σε κάθε batch ξεπερνούν ένα threshold, οπότε εφαρμόζεται ο k-median στα αποθηκευμένα medians. Ο αλγόριθμος αντιμετωπίζει το πρόβλημα της περιορισμένης μνήμης και την εφαρμογή των πράξεων σε ένα πέρασμα, όμως δεν αντιμετωπίζει τα outliers, ούτε λαμβάνει υπ'όψιν το πέρασμα του χρόνου, με αποτέλεσμα όλα τα δεδομένα από την αρχή του stream να έχουν την ίδια βαρύτητα στον υπολογισμό των clusters. Ο αλγόριθμος CluStream [1] , έπειτα, λειτουργεί με online-offline μοντέλο. Στην online φάση μετατρέπει τα raw δεδομένα σε microclusters, μία περίληψη των raw δεδομένων, καθιστώντας τα σημαντικά μικρότερα από τα raw δεδομένα. Στην offline φάση ο χρήστης ορίζει ένα διάστημα χρόνου για να δει τα clusters που υπήρχαν τότε, εκτελείται ο k-means αλγόριθμος στα micro-clusters και τα αποτελέσματά του ονομάζονται macro-clusters. Ειδικότερα, ένα micro-cluster αποτελείται από το tuple $(CF2^x, CF1^x, CF2^t, CF1^t, n)$, όπου $CF2^x$ είναι το άθροισμα των τετραγώνων των σημείων σε όλες τις d-διαστάσεις και το $CF1^x$ το άθροισμά τους σε όλες τις d-διαστάσεις, τα $CF2^t, CF1^t$ είναι το άθροισμα των τετραγώνων και των απλών χρονικών στιγμών, αντίστοιχα, και το n είναι το πλήθος των σημείων. Επιπλέον, όταν ξεπεραστεί ένα συγκεκριμένο κατώφλι τα micro-clusters διαγράφονται αν είναι μη ενεργά, αλλιώς συγχωνεύονται τα πιο κοντινά μεταξύ τους και τα id τους διατηρούνται σε μια λίστα. Τα microclusters αποθηκεύονται, συνήθως στο δίσκο, ανά κάποιες χρονικές στιγμές και ονομάζονται snapshots, ενώ για να διατηρηθούν αυτά δημιουργείται η έννοια του pyramidal time frame, κατά το οποίο τα snapshots αποθηκεύονται σε διαφορετικό βαθμό ανάλογα με το πόσο πρόσφατα είναι. Έπειτα, ομαδοποιούνται με διαφορετικά orders, τα οποία μπορεί να έχουν τιμές από 1 έως το log(T), με T τον παρόντα χρόνο. Πιο συγκεκριμένα, το snapshot που ανήκει στο i-th order δημιουργήθηκε σε κάποια χρονική στιγμή t η οποία διαιρείται ακριβώς από το α, με το α να δηλώνεται από το χρήστη. Ο χρήστης μπορεί να δει τα

clusters που έχουν δημιουργηθεί μέσα σε ένα χρονικό διάστημα δηλώνοντας μια τιμή h, ο αλγόριθμος ελέγχει ποια micro-clusters ήταν αποθηκευμένα στο χρόνο T-h μέχρι και το παρόν, διατηρεί τα micro-clusters που δημιουργήθηκαν μετά το T-h και σε αυτά εφαρμόζει των αλγόριθμο k-means. Τα αρνητικά του αλγορίθμου είναι πως δεν μπορεί να φτιάξει τυχαία σχήματα στα clusters, καθώς χρησιμοποιεί τον k-means αλγόριθμο και δεν αντιμετωπίζει τα προβλήματα του θορύβου, παρόλο που γίνεται διαγραφή των μη ενεργών micro-clusters ή αυτών που δεν πληρούν τις προϋποθέσεις για τη διατήρησή του.

Density-based αλγορίθμους, οι οποίοι θεωρούν ως clusters περιοχές στο χώρο με υψηλή συγκέντρωση δεδομένων (density), με ξεκάθαρη τη διαφορά τους από τις περιοχές με χαμηλό density. Μπορούν να δημιουργήσουν clusters με τυχαίο σχήμα και μπορούν να αντιμετωπίσουν αποτελεσματικά το θόρυβο χωρίς να είναι αναγκαίο ο χρήστης να προσδιορίσει τον αριθμό των clusters που θα δημιουργήσουν οι αλγόριθμοι, καθώς ως ορίσματα έχουν τη σχέση των δεδομένων με τους γείτονές τους και την πυκνότητα που θα πρέπει να έχει μια περιοχή ώστε να θεωρηθεί cluster. Data streaming clustreing αλγόριθμοι αυτής της κατηγορίας είναι ο DenStream [5], rDenStream [6], HDDStream [7]. Ο αλγόριθμος DenStream [5] δουλεύει με την χρήση dumped window model, δηλαδή η βαρύτητα κάθε σημείου που εισέρχεται από το stream μειώνεται εκθετικά με βάση το χρόνο βάσει της συνάρτησης $f(t) = 2^{-\lambda t}$. Το συνολικό βάρος του data stream προκύπτει από το $W = \frac{v}{1-2^{\lambda}}$, με υ τον αριθμό από τα δεδομένα που εισέρχονται βάσει μια μονάδας χρόνου. Προτείνονται δύο τύπου micro-clusters, ώστε να μπορούν να αλλάζουν περιοδικά με το χρόνο: το potential core-micro-cluster (p-micro-cluster) και το outlier-cmc. Τα p-cmc πρέπει να ξεπερνούν τη βαρύτητα $W \geq \beta * \mu$, με β το κατώφλι, ενώ τα outliers να είναι μικρότερα από τη βαρύτητα $W < \beta * \mu$. Ο αλγόριθμος έχει δυο φάσεις: την online, κατά την οποία διατηρεί τα cmc και την offline, κατά την οποία δημιουργεί clusters όταν τα ζητάει ο χρήστης. Ως προς τη διατήρηση των cmc, τα o-cmc διατηρούνται σε ένα buffer με τη λογική ότι τα καινούργια σημεία μπορούν να δημιουργήσουν ένα p-cmc. Η πρώτη συνάρτηση του αλγορίθμου είναι η συγχώνευση, δηλαδή, όταν ένα καινούργιο σημείο εισέρχεται, ελέγχεται αν αυτό πληροί τις προϋποθέσεις για να μπει στο πιο κοντινό p-cmc, με την προϋπόθεση ότι η καινούρια ακτίνα του cmc είναι μικρότερη από το ε, ενώ σε αρνητική περίπτωση ελέγχεται στο πιο κοντινό o-cmc αν η καινούρια ακτίνα του είναι μικρότερη του ε, οπότε και προστίθεται σε αυτό το o-cmc. Αντίθετα, σε καταφατική περίπτωση ελέγχεται αν το o-cmc έχει βαρύτητα μεγαλύτερη/ίση του $\beta * \mu$, ώστε να γίνει p-cmc, οπότε και βγαίνει από το buffer των outliers. Εάν πάλι τίποτα από τα παραπάνω δεν ισχύει, δημιουργείται ένα o-cmc με το σημείο και προστίθεται στο buffer των outliers. Επιπλέον, πρέπει να ελέγχεται ανά $T_p = \lceil \frac{1}{\lambda} * log(\frac{\beta_\mu}{\beta_{\mu-1}}) \rceil$, αν το βάρος κάθε p-cmc παραμένει μεγαλύτερο/ίσο του β*μ, καθώς μειώνεται χρονικά, ώστε ο αριθμός των p-cmc να παραμένει κάτω από $\frac{W}{\beta * \mu}$. Επίσης, πρέπει να διαχωρίζεται αν τα o-cmc θα μπορέσουν να γίνουν p-cmc ή αν είναι πραγματικά outliers, ελέγχοντας ανά Tp αν το βάρος κάθε o-cmc δεν ξεπερνάει τον αριθμό $\xi = \frac{2^{-\lambda(t_c - t_o + T_p)}}{2^{-\lambda T_p}}$, οπότε είναι outlier και

άρα θα πρέπει να διαγραφεί από το buffer των outliers. Για την αρχικοποίηση των πρώτων p-cmc εφαρμόζεται ο DBSCAN και έπειτα συνεχίζεται η παραπάνω διαδικασία. Στην offline διαδικασία, για να ευρεθούν τα clusters που υπάρχουν κάποια χρονική στιγμή, εφαρμόζεται ο DBSCAN στα υπάρχοντα p-cmc με τη διαφορά ότι η απόσταση δυο p-cmc θα πρέπει να μην ξεπερνάει το άθροισμα των ακτινών τους. Τα density connected p-cmc αποτελούν τα clusters. Το αρνητικό του αλγορίθμου είναι πως οι πρώτοι παράμετροι που παίρνει μένουν σταθερές σε όλη τη διάρκεια του streaming, ενώ για πραγματικά streams θα έπρεπε να προσαρμόζονται ανάλογα από το density των δεδομένων που εισέρχονται.

3

Density Grids

Στη διαδικασία του online clustering το πρώτο πρόβλημα που προκύπτει είναι η διατήρηση του dataset στη μνήμη του υπολογιστή. Το ζήτημα εντοπίζεται στο γεγονός ότι επειδή τα δεδομένα εισέρχονται σε πραγματικό χρόνο, η αποθήκευσή τους ως έχουν είναι μη εφικτή λύση, καθώς, όσο περνάει ο χρόνος, τα δεδομένα αυξάνονται σε απαγορευτικό βαθμό για τη μνήμη του υπολογιστή. Επιπλέον, ούτε η αποθήκευσή τους σε σύστημα αρχείων ή σε μια βάση δεδομένων αποτελεί αποδεκτή λύση, επειδή θα κόστιζε υπολογιστικό χρόνο στον αλγόριθμο και θα αναιρούνταν η έννοια του Online.

Για την αντιμετώπιση αυτού του προβλήματος γίνεται διαίρεση του d-διάστατου χώρου των δεδομένων σε density grids, με κάθε δεδομένο που εισέρχεται από το stream νσ ανατίθεται στο αντίστοιχο density grid και έπειτα να διαγράφεται. Με αυτό τον τρόπο χάνεται η ακρίβεια των αποτελεσμάτων, ωστόσο είναι εφικτή η διαδικασία του online clustering, ανεξάρτητα από τη διάρκεια του stream και τον όγκο των δεδομένων.

Οι τύποι και οι ορισμοί που ακολουθούν είναι βασισμένοι στο [8].

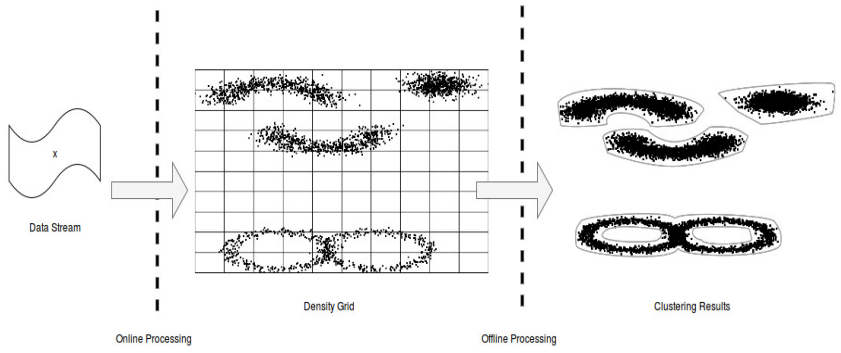

Fig. 3.1 Παράδειγμα χρήσης των grids.

3.1 Βασικοί ορισμοί

Κάθε δεδομένο που εισέρχεται από το stream θεωρείται ότι αποτελείται από d διαστάσεις και ορίζεται στο χώρο ως $S = S_1 \times S_2 \times ... \times S_d$ όπου S_i αντιπροσωπεύει την i^{η} διάσταση.

Για να διαιρεθεί το dataset S σε density grids χωρίζεται η κάθε δίασταση του σε p_i partitions , $S_i = S_{i,1} \cup S_{i,2} \cup ... \cup S_{i,p_i}$ με αποτέλεσμα να έχει διαχωριστεί συνολικά σε

$$N = \prod_{i=1}^{d} p_i \tag{3.1}$$

partitions.

Για κάθε δεδομένο x που έχει αντιστοιχιθεί σε ένα grid, ορίζεται ένας density συντελεστής που μειώνεται σύμφωνα με το χρόνο. Συγκεκριμένα για το χρόνο t_c, θεωρείται το time stamp $T(x) = t_c$, και ο συντελεστής density

$$D(x,t) = \lambda^{t-T(x)} = \lambda^{t-t_c} \tag{3.2}$$

όπου

$$\lambda \in (0,1) \tag{3.3}$$

ονομάζεται decay factor.

Grid Density : Για ένα grid g, μια χρονική στιγμή t, έστω ότι $E(g,t)$ είναι το σύνολο όλων των δεδομένων που έχουν ορισθεί ότι ανήκουν στο g μέχρι τη στιγμή t, τότε το density του grid g ορίζεται ως το άθροισμα όλων τον density συντελεστών όλων των δεδομένων που ανήκουν στο g.

$$D(g,t) = \sum_{x \in E(g,t)} D(x,t) \tag{3.4}$$

Αν και το density για κάθε grid αλλάζει συνέχεια με το χρόνο, για να υπολογιστεί για κάθε ένα grid σε όλη τη διάρκεια του streaming θα απαιτούσε μεγάλο υπολογιστικό κόστος και χώρο στην μνήμη. Για αυτό, ένας πιο αποτελεσματικός τρόπος για την διατήρησή αυτής της πληροφορίας είναι το density για κάθε grid να ενημερώνεται μόνο όταν ένα καινούριο δεδομένο αντιστοιχεί σε αυτό.

Έστω ότι σε ένα grid αντιστοιχεί ένα καινούριο δεδομένο την χρονική στιγμή t_n και η τελευταία φορά που έλαβε δεδομένο ήταν t_l τότε το density του grid μπορεί να υπολογισθεί με τον εξής τύπο :

$$D(g,t_n) = (\lambda^{t_n - t_l}) \cdot D(g,t_l) + 1 \tag{3.5}$$

Σύμφωνα με το (3.5) δεν είναι πλέον απαραίτητη η αποθήκευση ούτε των time stamps ούτε των density των δεδομένων που αντιστοιχούν σε ένα grid.

Οι απαραίτητες τιμές που χρειάζεται το κάθε grid μπορούν να διατηρηθούν σε ένα διάνυσμα που ονομάζεται Characteristic Vector.

Characteristic Vector : Το διάνυσμα ενός grid αποτελείται από ένα tuple με τις εξής μεταβλητές:

- t_g: η χρονική στιγμή όπου το grid ενημερώθηκε τελευταία φορά.
- t_m: η χρονική στιγμή όπου το grid διαγράφθηκε από την grid list.
- D: το density του grid από την τελευταία ενημέρωση του.
- label: το όνομα της κλάσης που ανήκει το grid
- status: flag (SPORADIC/NORMAL) για την διαγραφή των sporadic grids.

Η λειτουργία του Characteristic Vector αναλύεται σε επόμενο κεφάλαιο.

3.2 Clusters με βάση το density

Κάθε grid g στο χώρο ανάλογα με το density που έχει μια χρονική στιγμή t ανάγεται σε μια κατηγορία από τις τρεις : Dense, Transitional και Sparse.

Ένα grid είναι dense όταν το density του για μία χρονική στιγμή t είναι μεγαλύτερο από το threshold

$$D(g,t) > \frac{C_m}{N(1-\lambda)} = D_m \tag{3.6}$$

όπου $N > C_m > 1$.

Ένα grid είναι sparse όταν το density του για μία χρονική στιγμή t είναι μικρότερο από το threshold

$$D(g,t) < \frac{C_l}{N(1-\lambda)} = D_l \tag{3.7}$$

όπου $0 > C_l < 1$.

Ένα grid είναι transitional όταν ανήκει στο κλειστό διάστημα $[C_l, C_m]$. Οπότε για να είναι ένα grid transitional πρέπει να ικανοποιείται ο τύπος: Ένα grid είναι sparse όταν το density του για μία χρονική στιγμή t είναι μικρότερο από το threshold

$$D_l \leq D(g,t) \leq D_m \tag{3.8}$$

Definition 3.1. Γειτονικά grids: Για δύο density grids $g_1 = (j_1^1, j_2^1, ..., j_d^1)$ και $g_2 = (j_1^2, j_2^2, ..., j_d^2)$ στο d-διάστατο χώρο αν υπάρχει k, με, όπου ικανοποιεί τις εξής συνθήκες :

- $j_i^1 = j_i^2$ με $i = 1, 2, \ldots, k-1, k+1, \ldots, d$
- $|j_k^1 - j_k^2| = 1$

Τότε τα g_1 και g_2 είναι γειτονικά grids στην k^η διάσταση και αυτό συμβολίζεται ως $g_1 \sim g_2$.

Definition 3.2. Ομάδα grid: Ένα σύνολο από density grids $G = (g_1, ..., g_m)$ αποτελεί ομάδα grid αν για οποιαδήποτε δύο $g_i, g_j \in G$, υπάρχει μια ακολουθία με grids (group grid) τέτοια ώστε το $g_{k_1} = g_i$, $g_{k_l} = g_j$ και $g_{k_1} \sim g_{k_2}, g_{k_2} \sim g_{k_3}, ..., g_{k_l-1} \sim g_{k_l}$.

Definition 3.3. Εσωτερικά και Εξωτερικά Grids: Έστω μια ομάδα grid G και ένα grid όπου $g \in G$, αν το $g = (j_1, ..., j_d)$ έχει γειτονικά grids σε κάθε διάσταση από την 1^η μέχρι και την d^η τότε το g είναι εσωτερικό g στην ομάδα G. Σε αντίθετη περίπτωση, το g είναι εξωτερικό grid στην ομάδα G.

Definition 3.4. Grid Cluster: Με βάση τους τρεις προηγούμενους ορισμούς ορίζεται τι είναι το Cluster με βάση το density. Αν για μια ομάδα grid $G = (g_1, ..., g_m)$ ισχύει ότι κάθε εσωτερικό grid της είναι dense και κάθε εξωτερικό grid της είναι είτε dense είτε transitional, τότε το G αποτελεί ένα grid cluster. Από αυτόν τον ορισμό προκύπτει ότι ένα grid cluster αποτελείται από μια συνδεδεμένη ομάδα grid με υψηλότερο density από τα περιμετρικά της grids.

4

Στοιχεία του D-Stream

4.1 Interval gap

Το density για κάθε grid μειώνεται όσο περνάει ο χρόνος περιοδικά, με αποτέλεσμα τα grids που ήταν dense, αν δεν έχουν λάβει καινούρια δεδομένα για μεγάλο χρονικό διάστημα, να μειωθεί τόσο το density τους, ώστε να γίνουν transitional ή sparse. Αντίστοιχα, ένα sparse grid εάν λάβει καινούρια δεδομένα και αυξηθεί το density του τόσο, έχει ως αποτέλεσμα να γίνει transitional ή και dense. Για αυτό, ανά κάποιο χρονικό διάστημα gap πρέπει να γίνεται έλεγχος στο density των grids και τα clusters, ώστε να προσαρμόζονται στα καινούρια δεδομένα που έχουν εισέλθει από το stream. Αν το gap είναι πολύ μεγάλο δε θα μπορεί ο αλγόριθμος να προσαρμοστεί στη δυναμική εξέλιξη του data stream. Αντίθετα, αν είναι πολύ μικρό τότε θα γίνονται πολύ συχνά οι υπολογισμοί, με συνέπεια να μεγαλώνουν το υπολογιστικό κόστος του αλγορίθμου και να μην μπορεί να προσαρμοστεί στην ταχύτητα του data stream.

Ο ελάχιστος χρόνος που χρειάζεται ένα dense grid για να μετατραπεί σε sparse grid είναι: Προκειμένου να αποφευχθούν οι παραπάνω περιπτώσεις, υπολογίζεται ο ελάχιστος χρόνος που χρειάζεται να περάσει ώστε ένα dense grid να γίνει sparse grid και ο ελάχιστος χρόνος που χρειάζεται για να γίνει το αντίθετο, δηλαδή ένα sparse grid να γίνει dense grid. Τότε το gap παίρνει την ελάχιστη τιμή μεταξύ των δύο ελάχιστων χρόνων.

Ο ελάχιστος χρόνος που χρειάζεται ένα dense grid για να γίνει sparse grid είναι:

$$\delta_0 = \lfloor log_\lambda(\frac{C_l}{C_m}) \rfloor \tag{4.1}$$

Ο ελάχιστος χρόνος που χρειάζεται ένα dense grid για να γίνει sparse grid είναι:

$$\delta_1 = \lfloor log_\lambda(\frac{N - C_m}{N - C_l}) \rfloor \tag{4.2}$$

Επομένως, για να μπορεί ο αλγόριθμος να αναγνωρίσει οποιαδήποτε αλλαγή γίνει στα grids, από sparse σε dense ή dense σε sparse, το gap παίρνει για τιμή το μικρότερο χρόνο μεταξύ τους

$$\begin{aligned} gap &= min(\delta_0, \delta_1) \\ gap &= min(\lfloor log_\lambda(\frac{C_l}{C_m})\rfloor, \lfloor log_\lambda(\frac{N - C_m}{N - C_l})\rfloor) \\ gap &= \lfloor log_\lambda(max(\frac{C_l}{C_m}, \frac{N - C_m}{N - C_l}))\rfloor \end{aligned} \tag{4.3}$$

4.2 Αντιμετώπιση των outliers

Ένα βασικό πρόβλημα που προκαλείται από τη διαίρεση ενός dataset σε density grids είναι ο μεγάλος αριθμός των grids που δημιουργούνται, συγκεκριμένα σε high-dimensional δεδομένα. Διότι, αν κάθε διάσταση ενός d-διάστατου dataset διαιρεθεί σε 40 περιοχές, τότε δημιουργούνται 40^d grids τα οποία πρέπει να ενημερώνονται για το density τους ανά gap χρονικό διάστημα και να λαμβάνονται υπ'όψιν στους υπολογισμούς των clusters.

Επειδή τα clusters που σχηματίζονται από τον αλγόριθμο βρίσκονται σε περιοχές με μεγάλο density, προκύπτει η παρατήρηση πως τα δεδομένα δεν μπορεί να είναι απλωμένα σε όλα τα grids, αλλά μάλλον σε ένα μικρό υποσύνολο του grid space. Ως αποτέλεσμα αυτού, παρατηρείται ότι υπάρχει μεγάλος αριθμός grids που δεν έχουν λάβει ποτέ δεδομένα. Αυτό αντιμετωπίζεται με την χρήση του Characteristic Vector, καθώς μόνο τα grids που έχουν λάβει δεδομένα έχουν Characteristic Vector.

Με την εφαρμογή του αλγορίθμου, προκύπτει ότι δεν αρκεί μόνο η χρήση του Characteristic Vector, καθώς πρέπει να ληφθεί υπ'όψιν η ύπαρξή outliers δεδομένων τα οποία έχουν σημαντικό ρόλο στην απόδοση και την ποιότητα του αλγόριθμου. Επειδή, αν δεν αντιμετωπιστούν ως θόρυβος στο dataset και δεν διαγραφούν, με την πάροδο του χρόνου αντιστοιχούν σε grids τα οποία θα έπρεπε να είναι κενά, με αποτέλεσμα να συμπεριλαμβάνονται στις πράξεις για clustering. Αυτά τα grids ονομάζονται sporadic grids. Επειδή το stream μπορεί να εισάγει μεγάλο αριθμό δεδομένων για μεγάλο χρονικό διάστημα, ο αριθμός των sporadic grids μπορεί να φτάσει σε υψηλά επίπεδα έχοντας επιπτώσεις στην απόδοση και την ποιότητα του αλγορίθμου. Για το λόγο αυτό είναι κρίσιμο να ανιχνεύονται και να διαγράφονται τα sporadic grids καθ' όλη την διάρκεια του clustering.

Υποψήφια sporadic grids είναι τα sparse grids. Υπάρχουν δύο λόγοι για να χαρακτηριστεί ένα grid ως sparse. Ο πρώτος είναι το grid να ήταν κάποτε dense και να μην έχει λάβει για κάποιο διάστημα καινούρια δεδομένα, με αποτέλεσμα να έγινε sparse. Ο δεύτερος είναι να έχει λάβει πολύ λίγα δεδομένα και να μην ξεπέρασε ποτέ το threshold των sparse grid. Τα grids που ανήκουν στην τελευταία κατηγορία είναι πραγματικά sporadic και πρέπει να διαγραφούν, καθώς από

την πρώτη κατηγορία έχουν πολλές πιθανότητες να ξανά λάβουν δεδομένα και να αυξηθεί το density τους. Διαγράφοντας την πρώτη κατηγορία sparse grid μπορεί να προκαλέσει σημαντική μείωση στην ποιότητα του clustering.

Για να ορίσουμε τους κανόνες που διαχωρίζουν αυτές τις δύο κατηγορίες ορίζεται ένα threshold για το density. Αν η τελευταία φορά που έλαβε δεδομένα το grid g ήταν την χρονική στιγμή t_g, τότε για την χρονική στιγμή t, με $t > t_g$, το threshold για το density είναι:

$$\pi(t_g, t) = \frac{C_l}{N} \sum_{i=0}^{t-t_g} \lambda^i = \frac{C_l(1 - \lambda^{t-t_g+1})}{N(1-\lambda)} \tag{4.4}$$

Οπότε για να θεωρηθεί ένα sparse grid ως sporadic πρέπει να πλοιρούνται οι παρακάτω κανόνες:

- $D(g,t) < \pi(t_g, t)$ (S1)
- $t \geq (1+\beta)t_m$, όπου t_m είναι η χρονική στιγμή όπου διαγράφηκε ξανά το g και $\beta > 0$ μια σταθερά. (S2)

Το t_g και το t_m είναι αποθηκευμένα στο Characteristic Vector του κάθε grid.

Ο αλγόριθμος χρησιμοποιεί ένα hash table, το *grid_list*, για την αποθήκευση των grids τα οποία έχουν λάβει δεδομένα και ελέγχονται για το clustering. Ως κλειδιά έχει τις συντεταγμένες του κάθε grid και για τιμές έχει το Characteristic Vector του grid. Με το hash table είναι εφικτή η γρήγορη αναζήτηση, ανανέωση και η διαγραφή στα grids.

Οπότε, η έννοια της διαγραφής των sporadic grids εξισώνεται με τη διαγραφή του grid από το *grid_list* και την επαναφορά του density του στο μηδέν.

Η διαγραφή των sporadic grids γίνεται ανά gap χρόνο όταν ένα grid πληροί τους παρακάτω κανόνες:

- (D1) Αν ικανοποιεί τις σχέσεις (S1) και (S2), τότε στο Characteristic Vector του η μεταβλητή status σημειώνεται ως SPORADIC, όμως παραμένει στη λίστα μέχρι τον επόμενο έλεγχο.
- (D2) Αν ένα grid που έχει δηλωθεί ως SPORADIC δεν έχει καινούρια δεδομένα στο διάστημα που πέρασε από τον προηγούμενο έλεγχο, τότε διαγράφεται από τη λίστα *grid_list* και στο Characteristic Vector ενημερώνεται η τιμή t_m με τον τρέχοντα χρόνο. Σε αντίθετη περίπτωση, αν το grid συνεχίζει να ικανοποιεί τις σχέσεις (S1) και (S2), τότε παραμένει ως SPORADIC και ελέγχεται ξανά την επόμενη φορά, αλλιώς στο characteristic vector του η μεταβλητή status σημειώνεται ως NORMAL.

Τα διαγραμμένα grids μπορούν να ξανά προστεθούν στο *grid_list* αν υπάρξουν στο μέλλον καινούρια δεδομένα που αντιστοιχιστούν στο grid τους, όμως τα προηγούμενα δεδομένα που είχε δεν θα επηρεάσουν το clustering, καθώς έχουν διαγραφεί και το παλιό density του έχει μηδενιστεί. Αυτή η διαδικασία έχει ως αποτέλεσμα να διατηρείται ένα μικρό υποσύνολο των grids στην μνήμη

και να αποτρέπει την συσσώρευση απαγορευτικού αριθμού από sporadic grids στην μνήμη.

4.3 Διαδικασία Clustering

Ο αλγόριθμος διαχωρίζεται σε δυο στάδια το online και το offline.

Στο online στάδιο διαβάζει συνεχόμενα τα δεδομένα που εισέρχονται από το data stream και με την χρήση μιας hash συνάρτησης βρίσκει τις συντεταγμένες του grid στο οποίο ανήκει το κάθε δεδομένο. Αν το grid δεν έχει λάβει ακόμα δεδομένα, τότε αρχικοποιεί το Characteristic Vector του και εισάγεται στο *grid_list*. Αν υπάρχει ήδη το grid στην *grid_list*, τότε ανανεώνει το Characteristic Vector του grid, κάνοντας update την τιμή του density του με τον τύπο (3.5), με t_g την χρονική στιγμή που έλαβε το δεδομένο.

Στο offline στάδιο γίνεται δυναμικά η διαδικασία του clustering ανα gap χρονικές περιόδους. Μόλις περάσει το πρώτο gap διάστημα γίνεται η αρχικοποίηση των clusters και για το υπόλοιπο διάστημα ανά gap χρόνο πραγματοποιείται η προσαρμογή των ήδη υπάρχων clusters με τα καινούρια δεδομένα που έχουν εισέλθει από το stream.

Για την αρχικοποίηση των clusters, γίνεται update σε όλα τα density των grids που ανήκουν στο *grid_list*. Για το update των density γίνεται παραμετροποίηση του τύπου (3.5) ώστε να προσαρμόζεται το density με την πάροδο του χρόνου.

$$D(g, t_n) = (\lambda^{t_n - t_l}) \cdot D(g, t_l) \tag{4.5}$$

Αφού γίνει η ανανέωση των density σε όλα τα στοιχεία της *grid_list*, όλα τα grids που είναι dense grids εισάγονται σε μοναδικά clusters, με μοναδικό μέλος το grid αυτό. Στη συνέχεια, ελέγχονται οι γείτονες του κάθε cluster c ως προς το αν ανήκουν και αυτοί σε cluster ή όχι. Αν ανήκει ο γείτονας σε cluster c_h, τότε τα grids που ανήκουν στο μικρότερο cluster, από τα δύο που εξετάζονται, γίνονται merge στο μεγαλύτερο από τα clusters. Αν δεν ανήκουν σε κάποιο cluster οι γείτονες του cluster c που εξετάζεται, αλλά είναι transitional grids τότε και αυτά γίνονται μέλη στο cluster c.

Για την προσαρμογή των clusters, γίνεται update σε όλα τα density των grids που ανήκουν στο *grid_list* με τον τύπο (4.5). Μετά την ανανέωση των desnity σε όλα τα στοιχεία της *grid_list*, όποια grids δεν ανήκουν ήδη σε cluster εισάγονται σε μοναδικά clusters, με μοναδικό μέλος το grid αυτό. Έπειτα, όλα τα grids για τα οποία άλλαξε η κατάστασή τους στο density (sparse/dense/transitional) από τον προηγούμενο έλεγχο ανάλογα με το τωρινό τους density, ελέγχονται από συνθήκες ώστε να εισαχθούν σε καινούρια clusters ή να αφαιρεθούν από το cluster που ανήκουν.

Για κάθε grid με χαμηλό density που ανήκει στην κατηγορία sparse, αυτό διαγράφεται από το cluster που ανήκει και δεν ανήκει πλέον σε cluster. Στο

cluster στο οποίο άνηκε γίνεται έλεγχος αν μετά την διαγραφή του το cluster έγινε unconnected, δηλαδή δεν είναι πλέον ομάδα grid (3.2). Αν έχει γίνει unconnected, τότε το cluster διασπάται σε καινούργια μικρότερα clusters, τα οποία είναι connected.

Για κάθε grid με υψηλό density που ανήκει στην κατηγορία dense, βρίσκεται ο γείτονας του h που ανήκει στο μεγαλύτερο cluster c_h και αυτός, ανάλογα με το density του συγκρίνεται με το density και το μέγεθος του cluster, αν ανήκει σε κάποιο, που ανήκει το grid το οποίο εξετάζεται.

Στην περίπτωση που ο γείτονας h, του grid g, είναι dense grid, ελέγχεται αν το g ανήκει ή όχι σε cluster. Αν δεν ανήκει τότε το g εισάγεται στο cluster c_h. Αν ανήκει, τότε το cluster με το μικρότερο μέγεθος διαγράφεται και τα μέλη του εισάγονται στο μεγαλύτερο cluster από τα δύο.

Στην περίπτωση που ο γείτονας h είναι transitional grid, ελέγχεται αν το g ανήκει ή όχι σε cluster. Αν το g δεν ανήκει σε cluster, για να γίνει εισαγωγή του g στο cluster c_h πρέπει να ισχύει η συνθήκη ότι το h θα είναι εξωτερικό grid αν γίνει εισαγωγή του g στο cluster c_h, σε αντίθετη περίπτωση δε γίνεται κάποια ενέργεια. Αν το g ανήκει σε cluster, τότε αν το cluster c είναι μεγαλύτερο του c_h, το h θα μεταφερθεί από το cluster c_h στο c.

Για κάθε grid g με μέτριο density που ανήκει στην κατηγορία transitional, βρίσκεται ο γείτονας του h που ανήκει στο μεγαλύτερο cluster c_h και αν ικανοποιείται η συνθήκη ότι το grid g θα είναι εξωτερικό grid αν εισαχθεί στο cluster c_h, τότε το g εισάγεται στο cluster c_h.

Εν συντομία, ο αλγόριθμος δημιουργεί clusters τα οποία είναι connected ομάδες grid (3.2) που αποτελούνται από grids με υψηλότερο density από τα grids που τα περιστοιχίζουν. Αξιοσημείωτο είναι πως, κάθε φορά που είναι εφικτό, τα clusters γίνονται merge μεταξύ τους ώστε να περιστοιχίζονται από sparse grids. Επίπλεον, ο έλεγχος που γίνεται για τα transitional grids πριν την εισαγωγή τους σε ένα cluster προκύπτει από τον ορισμό (3.3), που δηλώνει ότι σε ένα cluster τα transitional grid που έχει ως μέλη είναι πάντα εξωτερικά grids. Στην περίπτωση που γίνουν merge δυο clusters, το μεγαλύτερο επικρατεί, με το μικρότερο να διαγράφεται και τα μέλη του να ενσωματώνονται στο μεγαλύτερο cluster. Στην εξέλιξη του streaming τα καινούργια grids που αποκτούν υψηλό density γίνονται clusters, ώστε να μπορούν να συμπεριληφθούν στη διαδικασία του clustering, γιατί διαφορετικά θα έπρεπε να ήταν κοντά σε κάποιο ήδη υπάρχον cluster που δημιουργήθηκε κατά την αρχικοποίηση των clusters.

5

Υλοποίηση

Στο Figure 1 απεικονίζεται το Flow Chart του αλγορίθμου D-Stream. Παρακάτω γίνεται ανάλυση των μεταβλητών και των συναρτήσεων που χρησιμοποιούνται από τον αλγόριθμο.

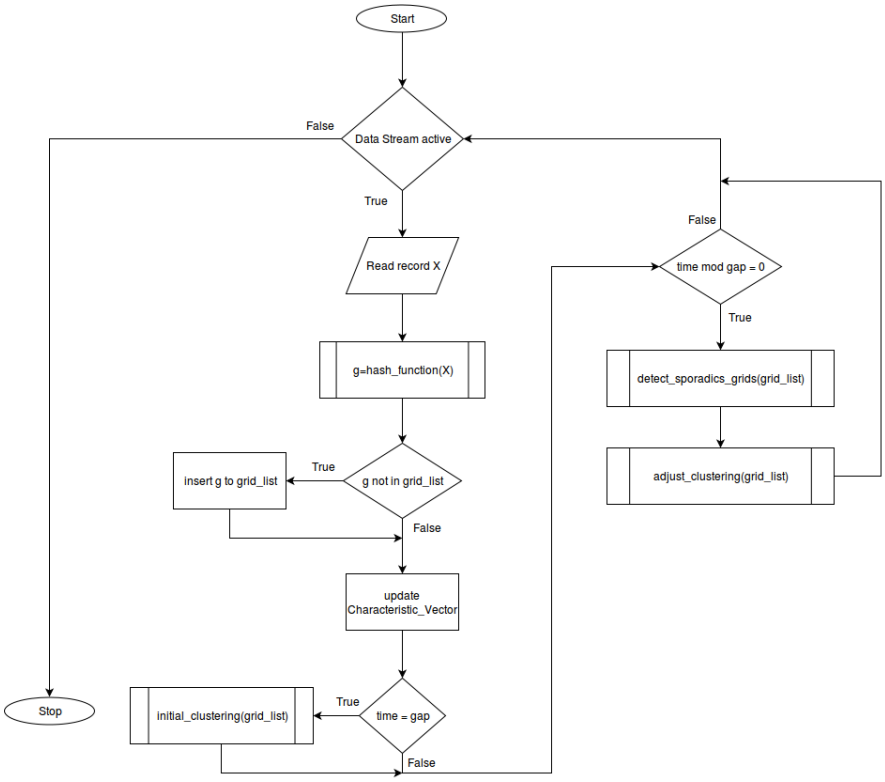

Fig. 5.1 Flow Chart του D-Stream.

5.1 Μεταβλητές

Ο αλγόριθμος D-Stream παίρνει σαν όρισμα τις εξής μεταβλητές:

- $decay\_factor$: Βαθμός μείωσης βαρύτητας των δεδομένων.
- cell: Λίστα με το μέγεθος των κελιών για κάθε διάσταση.
- cm: Παράμετρος που ορίζει το threshold για τα density grids.
- ratecm: Ποσοστό του N (γινόμενο όλων των partitions στα οποία έχει χωριστεί ο d-δίαστατος χώρος).
- cl: Παράμετρος που ορίζει το threshold για τα sparse grids.
- beta: Σταθερά που χρησιμοποιείται στην ανίχνευση των sporadic grids.
- $dimensions_limits$: Η μέγιστη και η ελάχιστη τιμή που μπορεί να πάρει το dataset που εξετάζεται σε κάθε διάσταση.
- ztime : Flag που ορίζει αν το dataset περιέχει δεδομένα UNIX epochs.

Η μεταβλητή cell είναι μια λίστα με στοιχεία ίσα με τον αριθμό των διαστάσεων του dataset και το κάθε στοιχείο της ορίζει το μέγεθος του κελιού της αντίστοιχης διάστασης, ώστε να είναι εφικτή η υλοποίηση του αλγορίθμου σε καινούρια datasets με διαφορετικές διαστάσεις δυναμικά.

Η μεταβλητή $decay\_factor$ ανήκει στο ανοιχτό διάστημα (0, 1). Επηρεάζει τις τιμές των threshold για density και sparse grids και το χρόνο gap, με τον αλγόριθμο περιοδικά ανά χρόνο gap να δημιουργεί clusters και να διαγράφει τα sporadic grids.

Η μεταβλητή cm ανήκει στο ανοιχτό διάστημα (1, N), με $N = \prod_{i=1}^{d}$ το γινόμενο των partitions στα οποία έχει χωριστεί ο d-δίαστατος χώρος.

Η μεταβλητή ratecm δηλώνει πόσο ποσοστό από το N θα οριστεί στο Cm. Το Cm, όπως πειραματικά έχει παρατηρηθεί, πρέπει να παίρνει τιμές κοντά στο N, ώστε το gap να παίρνει τιμές πάντα μεγαλύτερες του 1. Επειδή για διαφορετικά cell sizes ο d-διάστατος χώρος διαιρείται σε διαφορετικά partitions, αυτό έχει ως αποτέλεσμα το N να αλλάζει τιμές, ενώ με το rateCm εξασφαλίζεται ότι οι τιμές του Cm θα είναι πάντα κοντά στο N.

$$ratecm = 1 - ratecm$$

$$cm = N - (N * ratecm)$$

Η μεταβλητή cl ανήκει στο ανοιχτό διάστημα (0, 1).

Η μεταβλητή ztime είναι flag που δηλώνει αν το dataset περιέχει δεδομένα τύπου UNIX epochs. Οι τιμές που μπορεί να πάρει είναι είτε μικρότερες του 0, ώστε το flag να είναι False, είτε ίδιες με τον αριθμό της στήλης που ανήκουν

οι τιμές με UNIX epochs. Το flag υπάρχει ώστε ο χρήστης να δίνει μόνο ανά πόσες μέρες θέλει να χωριστεί η διάσταση του χρόνου (π.χ. $\frac{1}{4}$, $\frac{1}{2}$, 1 κ.ο.κ.).

5.2 Στάδια αλγορίθμου

Στον constructor του αλγορίθμου με βάση τις μεταβλητές που ορίζει ο χρήστης, υπολογίζονται οι εξής μεταβλητές :

- partitions: Ένα tuple με τον αριθμό των partitions στα οποία χωρίστηκε η κάθε διάσταση.
- N: $N = \prod_{i=1}^{d}$ το γινόμενο των partitions στα οποία έχει χωριστεί ο d-διάστατος χώρος.
- Dl: $D_l = \frac{C_l}{N(1-\lambda)}$ και $0 < Cl < 1$.
- Dm: $D_m = \frac{C_m}{N(1-\lambda)}$ και $1 < Cm < N$..
- gap: gap = min δ0,δ1 με
  $\delta_0 = \lfloor log_\lambda(\frac{C_l}{C_m}) \rfloor$ ο χρόνος που χρειάζεται ένα dense grid για να γίνει sparse.
  $\delta_1 = \lfloor log_\lambda(\frac{N-C_m}{N-C_l}) \rfloor$ ο χρόνος που χρειάζεται ένα sparse grid για να γίνει dense.

Κάθε δεδομένο που εισέρχεται από το data stream περνάει από την $hash\_function$, η οποία επιστρέφει τις συντεταγμένες του grid στις οποίες ανήκει το δεδομένο. Οι συντεταγμένες κάθε διάστασης υπολογίζονται με τον τύπο:

$$\frac{X_i}{Delta_i} * p_i + -lower\_index_i \tag{5.1}$$

X_i είναι το δεδομένο που εξετάζεται.
p_i είναι η τιμή του αριθμού με τον οποίο έχει διαιρεθεί η διάσταση i.
$Delta_i$ είναι η διαφορά της μέγιστης και της ελάχιστης τιμής της διάστασης i.
$lower\_index_i$ είναι οι συντεταγμένες του μικρότερου στοιχείου για τη διάσταση i του dataset. Αν αυτή είναι αρνητική βάζουμε +, σε αντίθετη περίπτωση −, ώστε η αρίθμηση να ξεκινάει από το 0.

Έπειτα, ελέγχεται αν το grid στο οποίο ανήκει το καινούριο δεδομένο υπάρχει στο $grid\_list$. Αν δεν υπάρχει, τότε γίνεται η αρχικοποίησή του με key τις συντεταγμένες του grid και value το διάνυσμα Characteristic Vector και εισάγεται στο $grid\_list$. Στην κλάση $Characteristic\_Vector$ αρχικοποιούνται οι μεταβλητές:

- $last\_time\_update\_of\_grid = 0$
- $last\_time\_remove\_of\_grid = 0$
- D = 0
- label = 'NO_CLASS'
- status = 'NORMAL'

- $density\_category$ = None
- $category\_changed\_last\_time$ = False

Στη συνέχεια ενημερώνεται το Characteristic Vector του αντίστοιχου grid για τα πεδία της πυκνότητας και του χρόνου κατά τον οποίο ενημερώθηκε τελευταία φορά το grid. Για την πυκνότητα εφαρμόζεται ο τύπος $D(g, t_n) = (\lambda^{t_n - t_l}) \cdot D(g, t_l) + 1$ και για το χρόνο ορίζεται το current time.

5.3 Πυκνότητα

Για να προσδιοριστεί αν ένα grid είναι dense/sparse/transitional χρησιμοποιείται η συνάρτηση $grid\_density\_category(density)$, η οποία παίρνει ως όρισμα το density του grid. Στη συνάρτηση γίνεται ελέγχος αν το D(x, t) είναι μεγαλύτερο/μικρότερο/ανάμεσα σε δύο thresholds που έχουν οριστεί για την κατηγοριοποίηση των grids και οι τύποι είναι οι παρακάτω:

- $D(x,t) > Dm$
- $D(x,t) < Dl$
- $Dl \leq D(x,t) \leq Dm$.

5.4 Γείτονες

Για την εύρεση των γειτόνων ενός grid χρησιμοποιείται η συνάρτηση $grid\_neighbors(grid)$, η οποία ακολουθεί την εξής διαδικασία: Πρώτα βρίσκονται για κάθε διάσταση οι υποψήφιοι γείτονες του grid που παίρνει ως όρισμα, με τη λογική ότι οι γείτονές του είναι "αριστερά και δεξιά" από τη συντεταγμένη της διάστασης που ελέγχεται. Έπειτα, από τους υποψήφιους γείτονες χαρακτηρίζονται ως γείτονες μόνο αυτοί που ανήκουν στο $grid\_list$.
Παράδειγμα σε 2-διάστατο χώρο:

	0	1	2
0		(0,1)	
1	(1,0)	(1,1)	(1,2)
2		(2,1)	

Έστω ότι το grid που ελέγχεται ως προς τους γείτονές του είναι το (1,1). Η έννοια "αριστερά και δεξιά" σημαίνει πως για τη διάσταση 1 (κάθετη) οι υποψήφιοι γείτονές του είναι τα grids : (1,0), (1,2) και για τη διάσταση 2 (οριζόντια) τα (0,1), (2,1).

5.5 Inside/outside grids

Με την συνάρτηση *inside_outside_grid*(*grid*) γίνεται ο προσδιορισμός ενός grid ως inside (εσωτερικό) ή outside (εξωτερικό) grid, ελέγχοντας αν έχει γειτονικά grids σε κάθε διάσταση, αλλιώς είναι outside grid αντίστοιχα.

5.6 Initial Clustering

Για τη δημιουργία των αρχικών clusters ακολουθείται η εξής διαδικασία : Αρχικά, για όλα τα grids που ανήκουν στην *grid_list* γίνεται ανανέωση των density τους με τη συνάρτηση *update_density_of_grids*(), η οποία χρησιμοποιεί τον τύπο (3.5) για τον υπολογισμό των καινούριων density.

Για κάθε dense grid δημιουργείται ένα cluster με το grid ως το μόνο στοιχείο του και όλα τα υπόλοιπα grids ορίζονται ως NO_CLASS. Τα ονόματα των clusters ορίζονται με την ίδια λογική που ακολουθούν τα auto increment keys στις βάσεις δεδομένων, δηλαδή ακέραιοι αριθμοί με κάθε καινούριο cluster να παίρνει την τιμή του τελευταίου συν ένα.

Σε μια επανάληψη, που σταματάει όταν δε γίνει καμία αλλαγή στα labels των grids, για κάθε cluster c εξετάζονται οι γείτονές του ως προς το αν είναι outside grids. Οι γείτονες που λαμβάνονται υπ' όψιν στη διαδικασία του initial clustering χωρίζονται σε δυο περιπτώσεις. Η πρώτη περίπτωση είναι ο γείτονας h να ανήκει σε κάποιο cluster c', οπότε το μικρότερο από τα δυο clusters ενσωματώνεται στο μεγαλύτερο. Δηλαδή:

- Αν $|c| > |c'|$ τότε $c \Leftarrow c \cup c'$
- Αν $|c| \leq |c'|$ τότε $c' \Leftarrow c' \cup c$

Η δεύτερη περίπτωση είναι ο γείτονας h να μην ανήκει σε κάποιο cluster, αλλά να είναι transitional grid, οπότε προστίθεται στο cluster c, $c \Leftarrow c \cup h$.

5.7 Adjust Clustering

Για την προσαρμογή των clusters στα καινούρια δεδομένα που εισέρχονται ανά gap χρονική περίοδο ακολουθείται η εξής διαδικασία:

Για όλα τα grids που ανήκουν στην *grid_list* γίνεται ανανέωση των density τους με τη συνάρτηση *update_density_of_grids*() και για κάθε dense grid που δεν ανήκει ήδη σε cluster δημιουργείται ένα cluster με το grid ως το μόνο στοιχείο.

Για κάθε grid που έχει αλλάξει κατηγορία (sparse/transitional/dense) από την τελευταία φορά που έγινε clustering εξετάζονται τρεις περιπτώσεις ανάλογα με την κατηγορία που ανήκει τώρα.

Αν το grid είναι sparse, τότε διαγράφεται το grid από το cluster c που ανήκει και ορίζεται ως NO_CLASS. Επιπλέον, ελέγχεται αν το cluster c παραμένει συνδεδεμένο (3.2). Αν δεν είναι, τότε δημιουργούνται καινούρια συνδεδεμένα clusters από τα υποσύνολα του αρχικού cluster.

Αν το grid g είναι dense, τότε από τους γείτονές του βρίσκεται το grid h που ανήκει στο μεγαλύτερο cluster c_h (το cluster με τα περισσότερα στοιχεία) και ανάλογα με την κατηγορία στην οποία ανήκει το h (transitional/dense) εξετάζονται δυο περιπτώσεις:

- Αν το h είναι dense τότε :
  - Αν το g δεν ανήκει σε κάποιο cluster (NO_CLASS), τότε ενσωματώνεται στο cluster του h, $c_h \Leftarrow c_h \cup g$
  - Αν το g ανήκει στο cluster c, ελέγχεται:
    - Αν $|c| > |c_h|$ τότε $c \Leftarrow c \cup c_h$
    - Αν $|c| \leq |c_h|$ τότε $c_h \Leftarrow c_h \cup c$
- Αν το h είναι transitional, τότε:
  - Αν το g δεν ανήκει σε κάποιο cluster (NO_CLASS) και το h είναι outside grid μετά την εισαγωγή του g στο cluster c_h, τότε το g μπαίνει στο cluster h
  - Αν το g ανήκει σε cluster c ελέγχεται:
    - Αν $|c| \geq |c_h|$ τότε $c \Leftarrow c \cup c_h$ και γίνεται έλεγχος αν το cluster c_h παραμένει συνδεδεμένο cluster. Αν δεν είναι, τότε δημιουργούνται καινούρια συνδεδεμένα clusters από τα υποσύνολα του αρχικού cluster, όπως και στην περίπτωση του sparse grid g.

Αν το grid είναι transitional, τότε από τα γειτονικά clusters επιλέγεται το cluster c' που ικανοποιεί τη συνθήκη:

- Αν το grid g θαείναι outside grid αν γίνει μέλος στο cluster, τότε $c' \Leftarrow c' \cup g$
  - Αν το grid g που μεταφέρθηκε ανήκε σε cluster c, τότε γίνεται έλεγχος για το αν παραμένει συνδεδεμένο το cluster c. Αν δεν είναι, τότε δημιουργούνται καινούρια συνδεδεμένα clusters από τα υποσύνολα του αρχικού cluster.

6

Πειραματική Αξιολόγηση

6.1 Ground Truth Clusters

Για την αξιολόγηση της ποιότητας του αλγορίθμου χρησιμοποιούνται δυο datasets: το πρώτο αποτελείται από συνθετικά δεδομένα και το δεύτερο από πραγματικά δεδομένα.

Τα συνθετικά δεδομένα κατασκευάζονται με τη χρήση μιας βιβλιοθήκης της γλώσσας προγραμματισμού python, την scikit-learn [9]. Ως ground truth clusters θεωρούνται τέσσερα σχήματα στο δισδιάστατο χώρο, τα οποία είναι 1 isotropic Gaussian blob, δύο εμπλεκόμενα ημικύκλια και δύο εφαπτόμενοι κύκλοι. Για το θόρυβο (outliers) προστίθενται τυχαία σημεία στο χώρο.

Για το πραγματικό data set χρησιμοποιείται το KDD Cup 1999 Data set [10]. Τα δεδομένα προέρχονται από το 1998 DARPA Intrusion Detection Evaluation Data και περιέχουν training data, τα οποία αποτελούνται από 7 εβδομάδων δεδομένα δικτύου με την κανονική ροή, αλλά και από δεδομένα από κακόβουλες συνδέσεις. Οι κατηγορίες των εισβολών είναι τέσσερις: Dos (Denial Of Service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to a local super-user privileges by a local un-privileged user), PROBING (surveillance and probing). Συνεπώς, για το dataset KDD Cup 1999, τα ground truth clusters είναι οι 4 κατηγορίες εισβολών και η (normal) κανονική ροή.

Για τα πειράματα που ακολουθούν, όλα τα datasets γίνονται normallize στο κλειστό δίαστημα [0,1]. Κάθε διάσταση διαιρείται σε ίσα κομμάτια, το καθένα από τα οποία έχει μήκος όσο η μεταβλητή cell.

6.2 Measures

Για την αξιολόγηση του αλγορίθμου χρησιμοποιούνται δυο measures, το purity και το SSQ.

6.2.1 Purity

Το purity υπολογίζει ένα score για τη διαδικασία του clustering που εφαρμόζεται σε ένα data stream. Το quality του clustering υπολογίζεται με από το μέσο purity των clusters.

$$purity = \frac{\sum_{i=1}^{K} \frac{|C_i^d|}{|C_i|}}{K} * 100 \tag{6.1}$$

Όπου K, δηλώνει τον αριθμό των clusters. Με $|C_i^d|$ δηλώνει τον αριθμό των points που ανήκουν στην επικρατούσα κλάση, με τον περισσότερο αριθμό στοιχείων, μέσα στο cluster i. Και $|C_i|$ δηλώνει το συνολικό αριθμό των points που ανήκουν στο cluster i. Η καλύτερη τιμή είναι το 100.

Σύμφωνα με το [5], πρέπει να λαμβάνεται υπ' όψιν ότι το dataset αλλάζει κατά τη διάρκεια του χρόνου, οπότε θα πρέπει να οριστεί μια μεταβλητή horizon (ή window). Άρα, το purity υπολογίζεται μόνο για τα στοιχεία που φτάνουν μέσα στο horizon.

6.2.2 SSQ

Μετράει το cohesiveness (συνεκτικότητα) των clusters υπολογίζοντας το sum of the square of distance για κάθε σημείο που έφτασε μέσα στο horizon με το centroid από το πιο κοντινό cluster [1]. Οι πιο μικρές τιμές είναι οι καλύτερες.

Το horizon ορίζεται με τον ίδιο τρόπο όπως και στο purity.

$$SSQ = \sum_{i=1}^{n} min(d^2(i, c_j)) \tag{6.2}$$

n: points που ανήκουν στο horizon.
$j = 1, \ldots, c$: όπου c ο αριθμός των clusters.
$d(i, c_j)$: η απόσταση του i^th point από το centroid c_j.

6.3 Αποτελέσματα

Πειραματικά βρέθηκαν τιμές για τις οποίες ο αλγόριθμος αντιμετωπίζει το θόρυβο και αναγνωρίζει αποτελεσματικά τα clusters για κάθε dataset. Για την προσομοίωση του streaming θεωρείται ότι η σειρά που έχουν γραφεί τα points στο dataset είναι και η σειρά δημιουργίας τους στο χρόνο. Επίσης, ορίζεται μια μεταβλητή *stream_speed*, που δηλώνει πόσα δεδομένα εισέρχονται από το stream ανά δευτερόλεπτο.

6.3.1 Θόρυβος

Για να αποδειχθεί ότι ο αλγόριθμος μπορεί να αντιμετωπίσει το θόρυβο, δημιουργείται ένα custom dataset με 30000 points και 10% θόρυβο. Τα δεδομένα είναι shuffled, ώστε να μη γίνει διαγραφή των clusters με το πέρασμα του χρόνου. Στο 6.1 απεικονίζονται τα μη επεξεργασμένα δεδομένα και τα clusters που δημιουργήθηκαν μετά την εκτέλεση του αλγορίθμου. Όπως φαίνεται, ο θόρυβος διαγράφεται αποτελεσματικά και τα τέσσερα clusters έχουν εντοπιστεί.

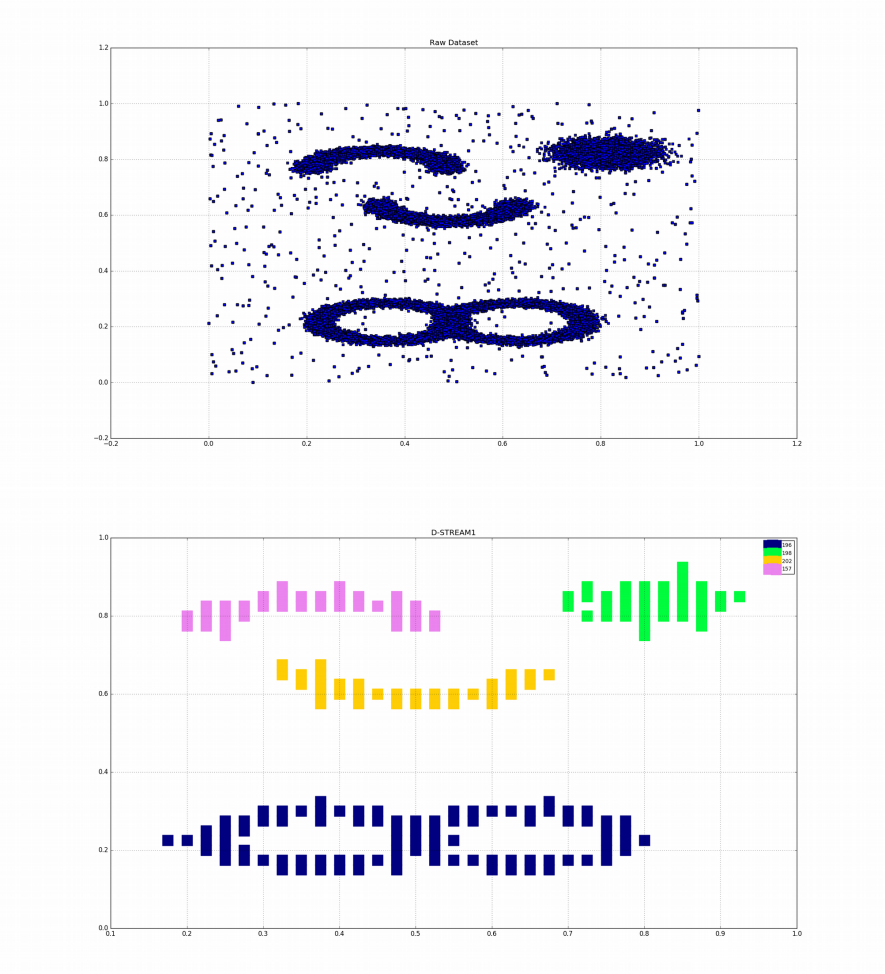

Fig. 6.1 Raw data και αποτελέσματα του clustering.

6.3.2 Clustering

Για να αποδειχθεί ότι ο αλγόριθμος προσαρμόζεται στη δυναμική εξέλιξη του dataset και μπορεί να αντιμετωπίσει το θόρυβο, δημιουργείται ένα dataset με 100K δεδομένα και θόρυβο 10% με 4 clusters σε σειριακή σειρά στο dataset. Ορίζεται το horizon h=5 και $stream\_speed = 1000$ και ελέγχεται η εξέλιξη του stream στις εξής χρονικές στιγμές: t_1=15s, t_2=36s, t_3=45s, t_4=55s, t_5=68, t_6=81s.Τα αποτελέσματα φαίνονται στα Fig 6.3 - 6.8.

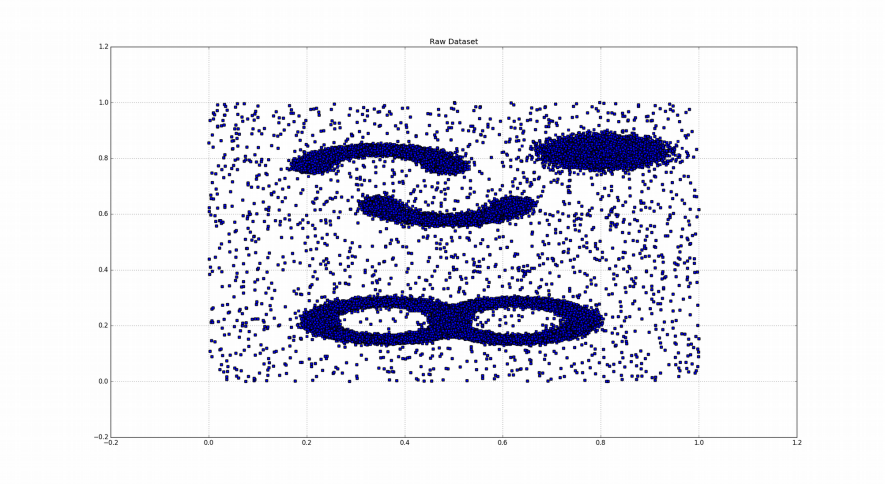

Fig. 6.2 100K raw data με 10% θόρυβο.

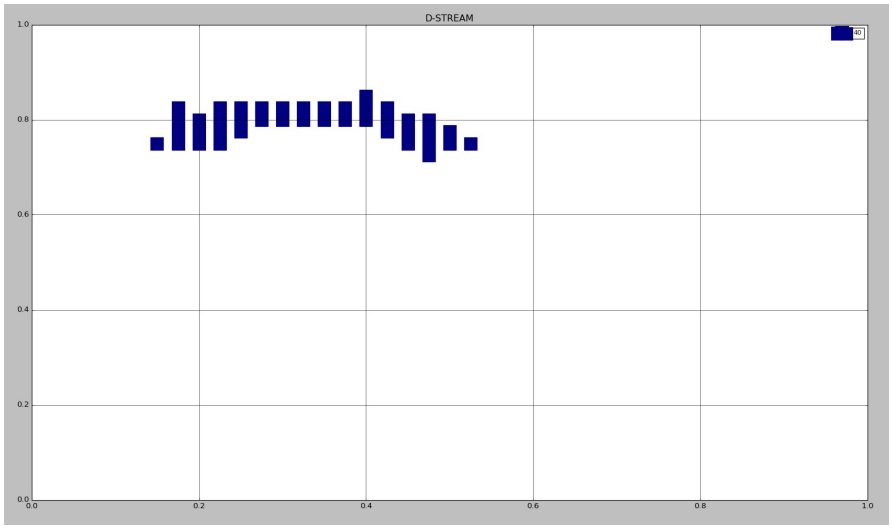

Fig. 6.3 Αποτελέσματα clustering τη χρονική στιγμή t=15s.

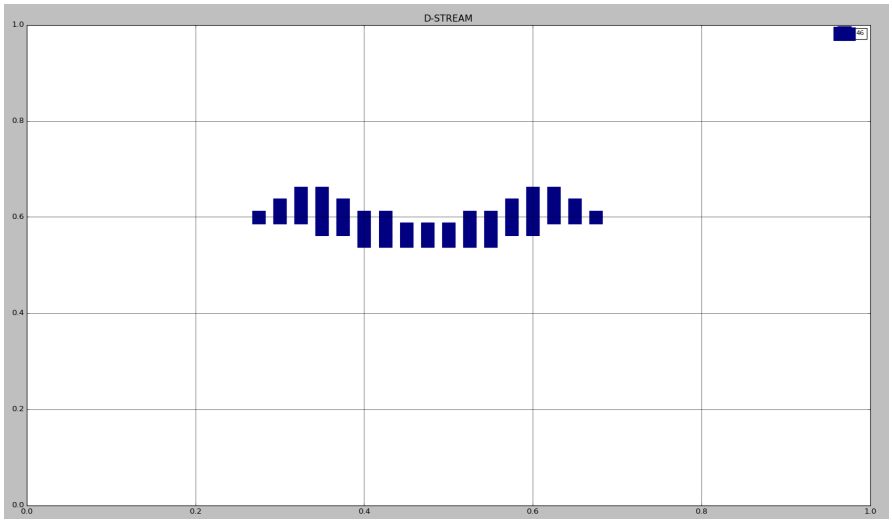

Fig. 6.4 Αποτελέσματα clustering τη χρονική στιγμή t=36s.

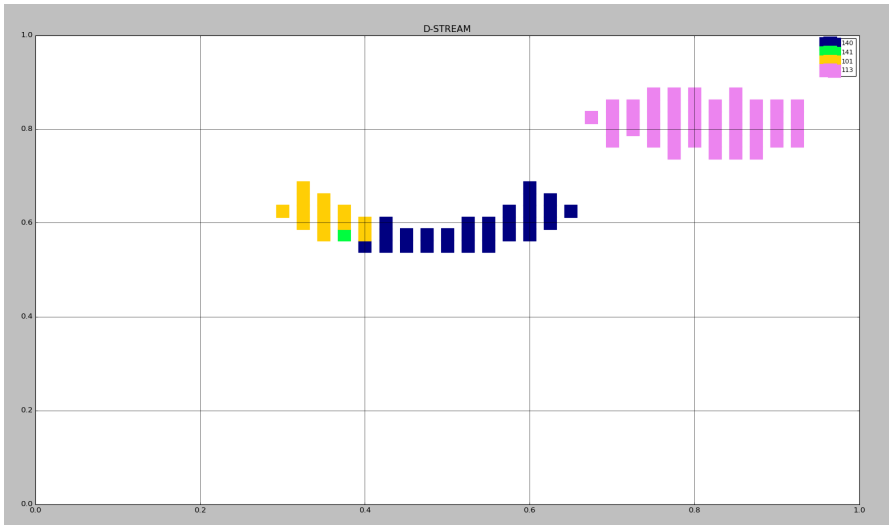

Fig. 6.5 Αποτελέσματα clustering τη χρονική στιγμή t=45s.

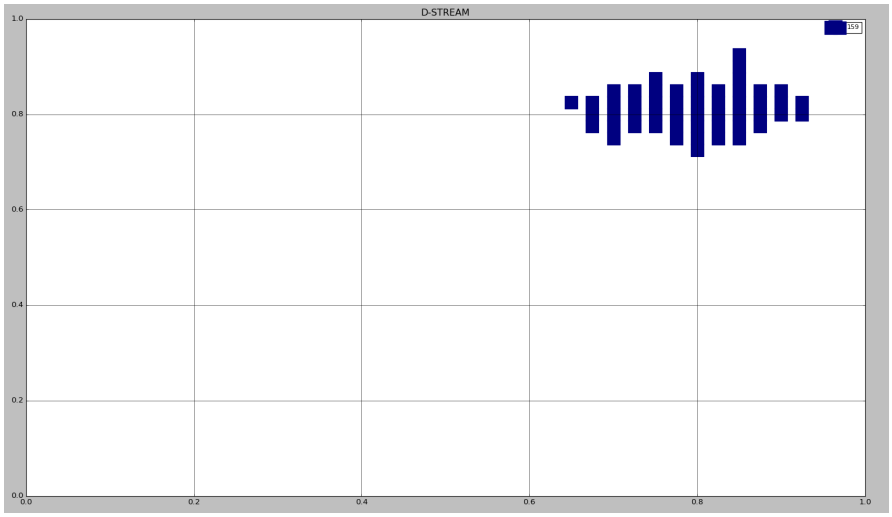

Fig. 6.6 Αποτελέσματα clustering τη χρονική στιγμή t=55s.

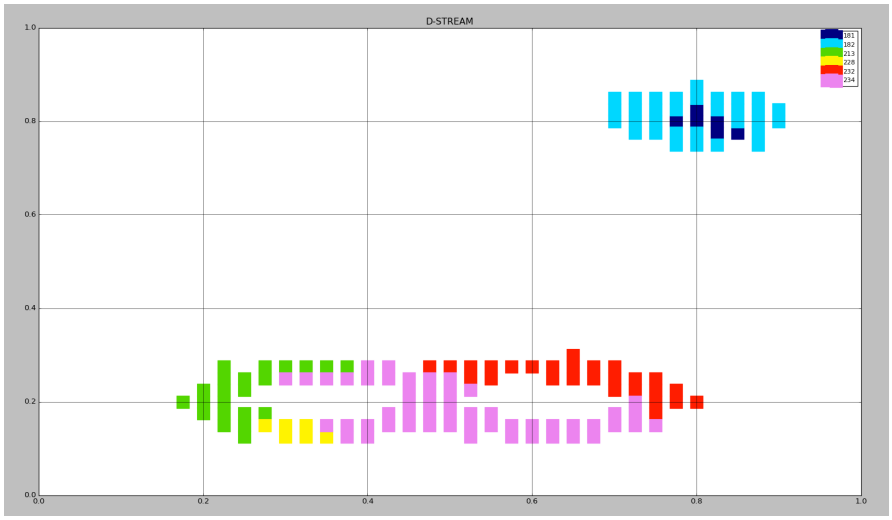

Fig. 6.7 Αποτελέσματα clustering τη χρονική στιγμή t=68s.

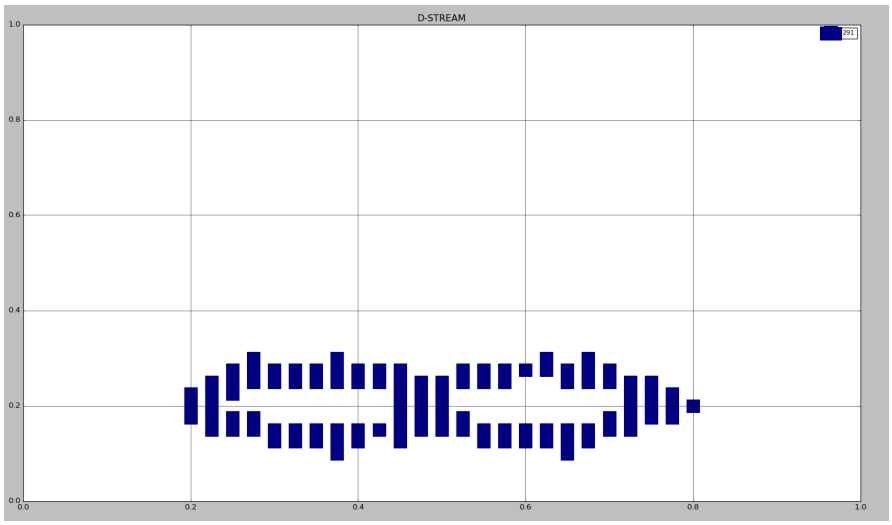

Fig. 6.8 Αποτελέσματα clustering τη χρονική στιγμή t=81s.

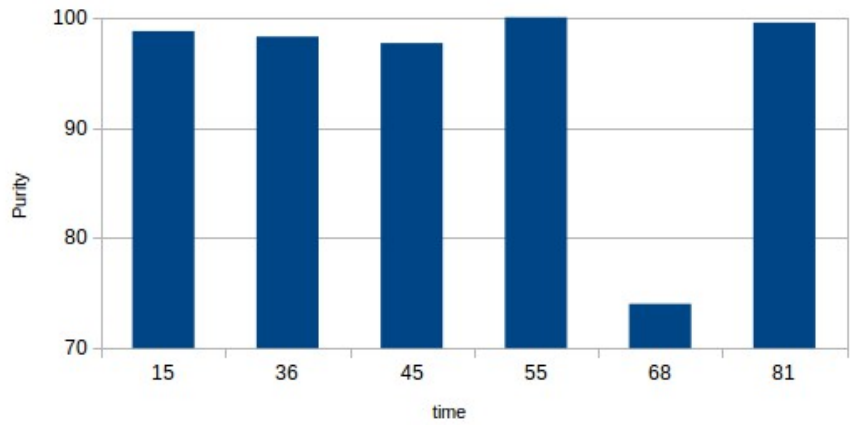

Fig. 6.9 Αποτελέσματα για το purity στο custom dataset.

Οι αξιολογήσεις του αλγορίθμου με βάση τα measures που αναφέρθηκαν παραπάνω είναι:

Από το 6.9 παρατηρείται πως το purity έχει συνεχώς υψηλές τιμές, εκτός από τις χρονικές στιγμές όταν ένα cluster αρχίζει να διαγράφεται, ενώ ένα άλλο αρχίζει να σχηματίζεται.

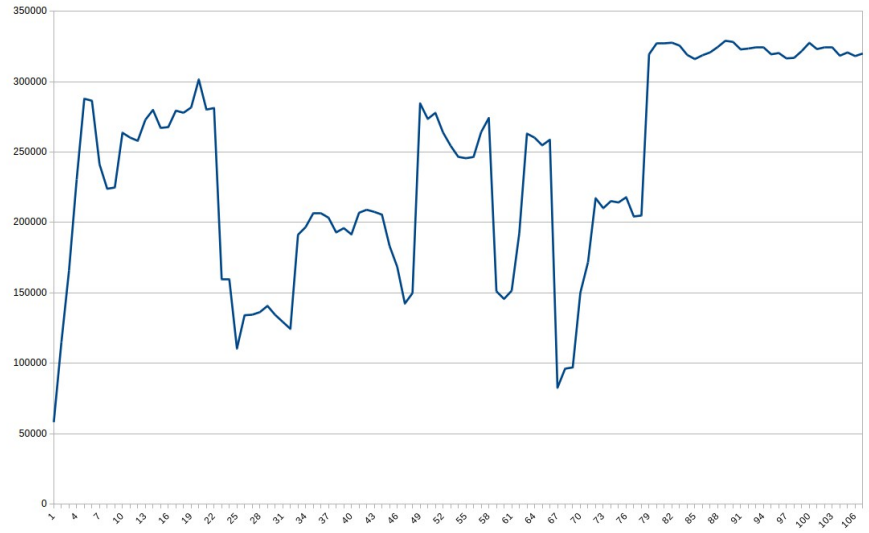

Fig. 6.10 Αποτελέσματα για το SSQ στο custom dataset.

Από το SSQ παρατηρούνται οι στιγμές που δημιουργούνται καινούρια clusters, επειδή γίνεται απότομη αλλαγή στις τιμές του. Επειδή το SSQ βασίζεται

στο centroid του κάθε cluster, οι τιμές του κυμαίνονται ανάλογα με το πόσο εκτεταμένο είναι στο χώρο.

6.3.3 KDD dataset

Το KDD dataset αποτελείται από 4 κλάσεις, με την κλάση normal να έχει το 19.69% των δεδομένων, την DOS το 79.24%, την Probe το 0,83%, την R2L το 0,23% και την U2R το 0,01%. Από αυτά τα δεδομένα προκύπτει ότι τα clusters που θα δημιουργηθούν θα παραμένουν σταθερά για αρκετό διάστημα και σε μικρές χρονικές στιγμές θα αλλάζουν από τις τρεις μικρότερες κλάσεις. Για την εφαρμογή και την αξιολόγηση του αλγορίθμου στο KDD dataset επιλέγονται δυο τιμές για τη διαίρεση των διαστάσεων σε grids. Τα αποτελέσματα που προκύπτουν για τις τιμές είναι τα παρακάτω:

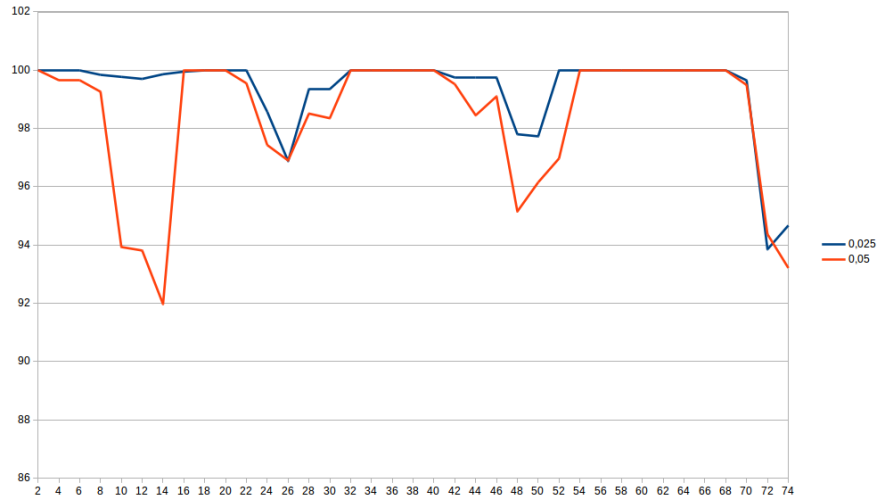

Fig. 6.11 Αποτελέσματα για το purity στο KDD dataset.

Από το 6.11 παρατηρείται πως αν η διάσταση χωριστεί σε μικρά grids, τότε αυξάνεται η ακρίβεια στο purity, καθώς τα grids που αποτελούν τα clusters μπορούν να διαχωρίσουν τις κλάσεις μεταξύ τους σε τυχόν κοντινές τιμές. Επειδή το dataset δεν έχει θόρυβο, αλλά μόνο τέσσερις κλάσεις, το purity έχει σταθερά υψηλές τιμές και μειώνεται μόνο όταν τοποθετούνται διαφορετικές κλάσεις στο ίδιο cluster. Αυτό δε συμβαίνει συχνά, λόγω των features που έχουν επιλεχθεί από το dataset, κάνοντας τη διαφοροποίηση των κλάσεων ξεκάθαρη.

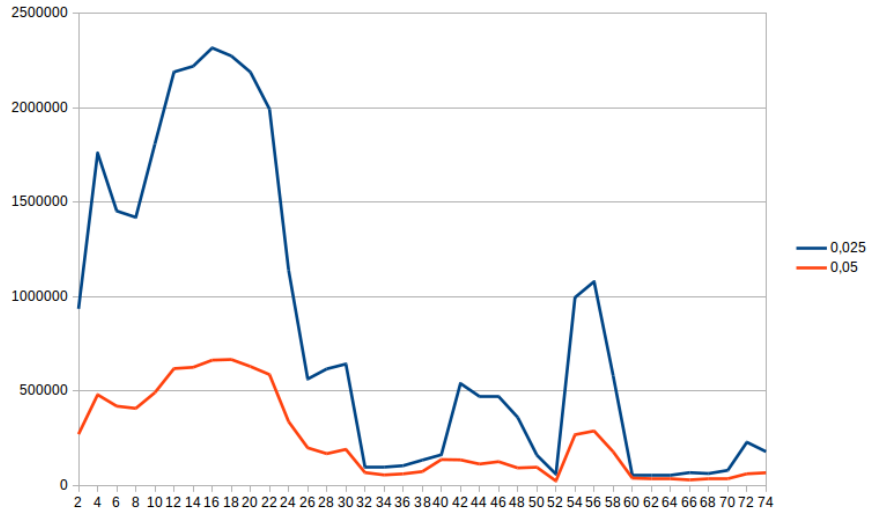

Fig. 6.12 Αποτελέσματα για το SSQ στο KDD dataset.

Από το 6.12 παρατηρείται πως αν η διάσταση χωριστεί σε μικρά grids, τότε το SSQ αυξάνεται. Αυτό προκύπτει από το γεγονός ότι τα clusters που δημιουργούνται έχουν μεγάλη έκταση στο χώρο. Παρατηρείται πως διατηρούνται πολύ χαμηλές τιμές για αρκετό διάστημα, εξαιτίας του γεγονότος ότι από το stream εισέρχονται δεδομένα από μια μόνο κλάση, με τις τιμές της να συγκεντρώνονται σε ένα συγκεκριμένο σημείο στο χώρο. Ενώ στις χρονικές στιγμές όπου το SSQ αυξάνεται απότομα, οι τιμές μεταφράζονται ως εισερχόμενες καινούριες κλάσεις από το stream.

Βιβλιογραφία

1. Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29, VLDB '03, 2003
2. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In Proceedings of the 41st Annual Symposium on Foundations of Computer Science. IEEE Computer Society, 2000.
3. Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. Knowledge and Information Systems, 15(2), 2008.
4. Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++: A clustering algorithm for data streams. ACM Journal of Experimental Algorithmics, 17(4), 2012.
5. Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In Proceedings of the Sixth SIAM Conference on Data Mining, 2006.
6. J. Gama and M. Gaber (Eds). Learning from Data Streams. Springer, 2007.
7. Irene Ntoutsi, Arthur Zimek, Themis Palpanas, Peer Kröger,and Hans-Peter Kriegel. Density-based projected clusteringover high dimensional data streams. In Proc. of the 12thSIAM International Conference on Data Mining, 2012.
8. Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.
9. Βιβλιοθήκη sklearn. http://scikit-learn.org/stable/index.html
10. KDD Cup 1999 http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html