ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προσαρμοστικό Σχήμα Συμπίεσης Δεδομένων σε Ασύρματα Δίκτυα Αισθητήρων

Μαρία Σ. Πουταχίδου

Δήμητρα Α. Τσιρίκου

Επιβλέποντες: **Ευστάθιος Χατζηευθυμιάδης,** Επίκουρος Καθηγητής

Χρήστος Αναγνωστόπουλος, Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2012

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Προσαρμοστικό Σχήμα Συμπίεσης Δεδομένων σε Ασύρματα Δίκτυα Αισθητήρων

Μαρία Σ. Πουταχίδου
Α.Μ.: Μ1151
Δήμητρα Α. Τσιρίκου
Α.Μ.: Μ1150

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ευστάθιος Χατζηευθυμιάδης,** Επίκουρος Καθηγητής
Χρήστος Αναγνωστόπουλος, Επίκουρος Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Λάζαρος Μεράκος,** Καθηγητής

Ιούλιος 2012

ΠΕΡΙΛΗΨΗ

Στόχος των σχημάτων συμπίεσης πληροφορίας πλαισίου είναι η μείωση των απαιτούμενων ενεργειακών πόρων, με παράλληλη διατήρηση της ποιότητας των παρεχόμενων υπηρεσιών σε υψηλό επίπεδο.

Στην εργασία αυτή, παρουσιάζονται τρία νέα σχήματα, που βασίζονται στο σχήμα PC3, για τη συμπίεση πληροφορίας πλαισίου προερχόμενης από ασύρματα δίκτυα αισθητήρων. Τα δύο πρώτα σχήματα χρησιμοποιούν τη θεωρία βέλτιστης παύσης για την εύρεση του χρονικού σημείου τερματισμού τη συμπίεσης. Το τρίτο σχήμα διαφέρει αρκετά από όλα τα προηγούμενα και χρησιμοποιεί την αυξητική ανάλυση κύριων συνιστωσών. Τα τρία αυτά σχήματα συγκρίνονται μεταξύ τους και με το PC3 ως προς το ενεργειακό κέρδος που επιτυγχάνουν, αλλά και ως προς το σφάλμα που έχουν κατά τη μετάδοση της πληροφορίας πλαισίου, με τα δυο πρώτα να επιτυγχάνουν σημαντική αύξηση του ενεργειακού κέρδους έναντι του PC3, χωρίς πολύ μεγάλη απώλεια πληροφορίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: επίγνωση πληροφορίας πλαισίου

ΛΕΞΕΙΣ-ΚΛΕΙΔΙΑ: συμπίεση, αυξητική ανάλυση κύριων συνιστωσών, θεωρία βέλτιστης παύσης, δίκτυα αισθητήρων

ABSTRACT

Context compression schemes aim to reduce the required energy resources retaining, in parallel, high quality provided services.

This thesis proposes three new schemes, which are based on the PC3 scheme, to compress context information derived from wireless sensor networks. The first two schemes make use of the optimal stopping theory to determine the proper time to pause the compression. The third one differs a lot from all the other schemes and uses the incremental principal component analysis. The three proposed schemes are compared with PC3 as to the energy gain they attain and also the error in the transmission of the context information. The first two schemes attain significant increase of the energy gain comparing with PC3, without losing much information.

SUBJECT AREA: Context aware information

KEYWORDS: compression, incremental principal component analysis, optimal stopping theory, wireless sensor networks

Στους γονείς μας,
που μας στήριξαν καθ’ όλη τη διάρκεια των σπουδών μας.

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό, επιθυμούμε να ευχαριστήσουμε τον κ. Ευστάθιο Χατζηευθυμιάδη και τον κ. Χρήστο Αναγνωστόπουλο για τον χρόνο που αφιέρωσαν και για τις αξιόλογες υποδείξεις τους, οι οποίες ήταν καθοριστικές για την ολοκλήρωση της διπλωματικής αυτής εργασίας.
Θα θέλαμε ακόμη να ευχαριστήσουμε τις οικογένειές μας για την αμέριστη συμπαράστασή και κατανόηση που έδειξαν καθ' όλη τη διάρκεια των προπτυχιακών και μεταπτυχιακών μας σπουδών.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Αυτή η διπλωματική εργασία είναι η τελευταία εργασία των συγγραφέων για την εκπλήρωση των υποχρεώσεων τους ως μεταπτυχιακές φοιτήτριες και τη λήψη του διπλώματος τους. Η μελέτη που απαιτήθηκε για την ανάπτυξη της, όπως επίσης και η τελική συγγραφή της, έλαβε χώρα στις Αχαρνές Αττικής και στο Ζωγράφο Αττικής, περιοχές διαμονής των συγγραφέων κατά το έτος 2012. Για την ολοκλήρωση της καθοριστικό ρόλο έπαιξαν οι συζητήσεις των συγγραφέων με τους επιβλέποντες, κ. Χατζηευθυμιάδη Ευστάθιο και κ. Αναγνωστόπουλο Χρήστο, τους οποίους οι συγγραφείς θα ήθελαν να ευχαριστήσουν ιδιαιτέρως για την αρωγή τους.

1. ΕΙΣΑΓΩΓΗ

Η πρόοδος στην τεχνολογία των μικρό-ηλεκτρομηχανικών συστημάτων (ΜΗΜΣ), στην ασύρματη επικοινωνία και στα ψηφιακά ηλεκτρονικά, ακολούθησε την ανάπτυξη στα συστήματα διάχυτου υπολογισμού. Αυτό έδωσε την δυνατότητα για την ανάπτυξη κόμβων αισθητήρων χαμηλού κόστους, χαμηλής κατανάλωσης ενέργειας και πολλών λειτουργιών, οι οποίοι είναι μικροί σε μέγεθος και επικοινωνούν, χωρίς ανθρώπινη παρέμβαση ή επιτήρηση, μεταξύ τους σε μικρές αποστάσεις. Αυτοί οι μικροσκοπικοί κόμβοι αισθητήρων, που αποτελούνται από εξαρτήματα ανίχνευσης, επεξεργασίας δεδομένων και επικοινωνίας, οδηγούν στην ιδέα δικτύων αισθητήρων που βασίζονται στην συνεργατική λειτουργία ενός μεγάλου συνόλου κόμβων. Τα δίκτυα αισθητήρων αντιπροσωπεύουν μια σημαντική εξέλιξη έναντι των παραδοσιακών αισθητήρων, οι οποίοι εγκαθίστανται με τους ακόλουθους δύο τρόπους [7] :

- Εγκατάσταση αισθητήρων μακριά από το πραγματικό φαινόμενο. Με αυτήν την προσέγγιση απαιτούνται μεγάλοι αισθητήρες που χρησιμοποιούν πολύπλοκες τεχνικές για να διακρίνουν τους στόχους από τον θόρυβο του περιβάλλοντος.
- Εγκατάσταση αισθητήρων που εκτελούν μόνο αισθητήρια εργασία. Η θέση και η επικοινωνιακή τοπολογία σχεδιάζονται προσεκτικά. Αυτοί εκπέμπουν μια σειρά του υπό παρακολούθηση φαινομένου στους κεντρικού κόμβους όπου εκτελούνται οι υπολογισμοί και συγχωνεύονται τα δεδομένα.

Ένα δίκτυο αισθητήρων αποτελείται από ένα μεγάλο αριθμό κόμβων αισθητήρων, οι οποίοι αναπτύσσονται πυκνά, είτε μέσα στο φαινόμενο-χώρο, που θέλουν να παρατηρήσουν, είτε πολύ κοντά σε αυτό. Η θέση των κόμβων αισθητήρων δεν είναι ανάγκη να προσχεδιαστεί ή να προαποφασιστεί. Αυτό επιτρέπει την τυχαία εξάπλωσή σε μη προσβάσιμα εδάφη ή σε επιχειρήσεις αντιμετώπισης καταστροφών. Από την άλλη πλευρά, αυτό σημαίνει ότι τα πρωτόκολλα και οι αλγόριθμοι των δικτύων αισθητήρων πρέπει να διαθέτουν αυτό-οργανωτικές δυνατότητες. Ένα άλλο μοναδικό χαρακτηριστικό των δικτύων αισθητήρων είναι η συνεργατική λειτουργία των κόμβων αισθητήρων. Οι αισθητήριοι κόμβοι εξοπλίζονται με έναν επεξεργαστή. Αντί να στέλνουν ακατέργαστα δεδομένα στους κόμβους που είναι υπεύθυνοι για την συγχώνευση των δεδομένων, οι αισθητήριοι κόμβοι χρησιμοποιούν τις δυνατότητες επεξεργασίας που διαθέτουν προκείμενου να εκτελέσουν τοπικά απλούς υπολογισμούς και να εκπέμψουν μόνο τα επεξεργασμένα δεδομένα. Το παραπάνω χαρακτηριστικό εξασφαλίζει ένα μεγάλο πλήθος εφαρμογών για δίκτυα αισθητήρων. Μερικές από τις περιοχές εφαρμογής είναι η υγεία, ο στρατός και η ασφάλεια. Για παράδειγμα, οι σωματικές μετρήσεις (πυρετός, αρτηριακή πίεση κτλ.) ενός ασθενή μπορούν να παρατηρούνται απομακρυσμένα από το γιατρό του. Αυτό είναι και πιο βολικό για τον ασθενή, αλλά επίσης επιτρέπει και στο γιατρό να έχει πλήρη εικόνα της παρούσας κατάστασης του ασθενή και της εξέλιξης της υγείας του. Οι αισθητήριοι κόμβοι μπορούν, επίσης, να χρησιμοποιηθούν για να εντοπίσουν ξένα χημικά στοιχεία στον αέρα και το νερό. Μπορούν να βοηθήσουν για να ανακαλύψουν τον τύπο, τη συγκέντρωση και τη θέση μολυσματικών ουσιών.

Προκειμένου να υλοποιηθούν οι παραπάνω, αλλά και άλλες εφαρμογές των δικτύων αισθητήρων, απαιτούνται τεχνικές ad-hoc. Παρόλο που αρκετοί αλγόριθμοι και πρωτόκολλα έχουν προταθεί για τα παραδοσιακά ad-hoc ασύρματα δίκτυα, δυστυχώς, δεν είναι δυνατόν να χρησιμοποιηθούν προκειμένου να καλύψουν τα μοναδικά χαρακτηριστικά και τις απαιτήσεις των εφαρμογών των δικτύων αισθητήρων.

Ένας από τους σημαντικότερους περιορισμούς στα δίκτυα ασύρματων αισθητήρων είναι η απαίτηση για χαμηλή κατανάλωση ενέργειας. Οι εκπομπές, οι λήψεις και η

επεξεργασία της πληροφορίας από τους αισθητήριους κόμβους, θα μπορούσαν να προκαλέσουν πολύ μεγάλη κατανάλωση της ενέργειας του συστήματος, σπαταλώντας έτσι την ενέργεια σε σύντομο χρονικό διάστημα. Όμως, η γρήγορη κατανάλωση ενέργειας είναι μία ανεπιθύμητη κατάσταση για πληθώρα αιτιών όπως λόγους οικονομικούς και εξοικονόμησης περιβαλλοντικών πόρων. Πρακτικότεροι λόγοι που καθιστούν ανεπιθύμητη τη μεγάλη δαπάνη ενέργειας είναι οι περιπτώσεις όπου η συντήρηση του συστήματος και ο ενεργειακός ανεφοδιασμός είναι εξαιρετικά δυσχερείς διαδικασίες, όπως όταν δίκτυα αισθητήρων τοποθετούνται σε απρόσιτες περιοχές με σκοπό την πυρανίχνευση. Αφού οι αισθητήριοι κόμβοι έχουν περιορισμένες και συνήθως αναντικατάστατες πηγές ενέργειας, ενώ τα παραδοσιακά δίκτυα στοχεύουν να παρέχουν υπηρεσίες υψηλής ποιότητας, τα δίκτυα ασύρματων αισθητήρων έχουν ως πρωταρχικό στόχο την διατήρηση της ενέργειας. Τα συστήματα που εξοικονομούν ενέργεια, ωστόσο, αναγκάζονται να εκπέμπουν την πληροφορία είτε επιλεκτικά, είτε συμπιεσμένη, είτε να την προβλέπουν σε ορισμένες περιπτώσεις με μαθηματικούς τρόπους. Το σημαντικότερο τίμημα στην προσπάθεια των συστημάτων αυτών για μείωση στην κατανάλωση ενέργειας είναι η αλλοίωση στην πληροφορία που επεξεργάζεται το σύστημα. Μεγάλη πρόκληση παραμένει, λοιπόν, η πρόταση σχημάτων τέτοιων που να οδηγούν σε λιγότερη κατανάλωση ενέργειας, χωρίς παράλληλα μεγάλη αύξηση του σφάλματος στη μεταφορά της πληροφορίας.

Στα πλαίσια της παρούσης διπλωματικής εργασίας, προτείνουμε τρία νέα σχήματα που βασίστηκαν στο σχήμα PC3 (Principal component–based context compression) [1] για τη συμπίεση πληροφορίας πλαισίου (contextual information) που προέρχεται από ασύρματα δίκτυα αισθητήρων, με στόχο τη χαμηλότερη κατανάλωση ενέργειας, με παράλληλη διατήρηση του σφάλματος στη μεταφορά της πληροφορίας σε ανεκτά επίπεδα. Τα δυο από τα τρία πρώτα σχήματα είναι εμπνευσμένα από τη θεωρία βέλτιστης παύσης (optimal stopping theory ή OST). Το τρίτο και τελευταίο σχήμα βασίζεται στην αυξητική ανάλυση κύριων συνιστωσών (incremental principal component analysis).

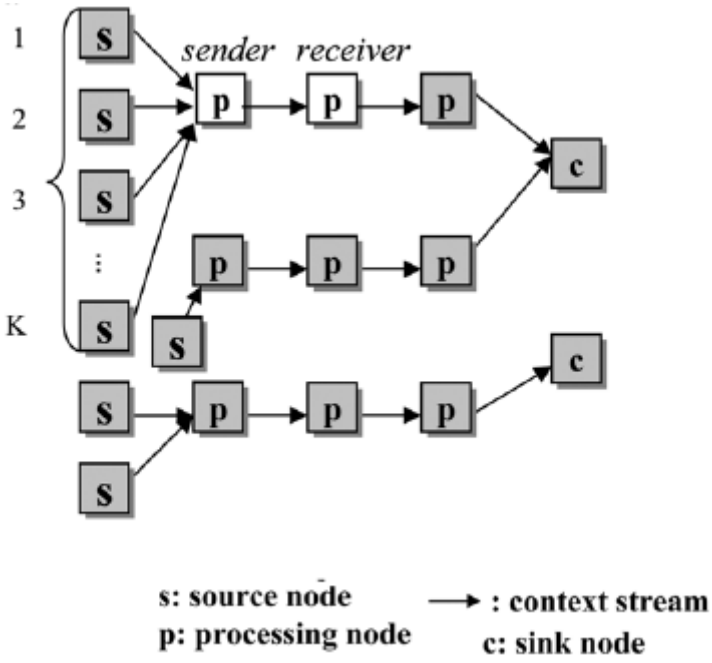
Οι υλοποιήσεις των αλγορίθμων των νέων προτεινόμενων σχημάτων, καθώς και τα πειράματα για την εκτίμηση της λειτουργίας τους, πραγματοποιήθηκαν στο περιβάλλον εργασίας MatlabR2009b.

2. PRINCIPAL COMPONENT – BASED CONTEXT COMPRESSION MODEL (PC3)

Στο κεφάλαιο αυτό θα περιγράψουμε τον αλγόριθμο Principal Component – based Context Compression Model (PC3) [1] από τον οποίο προέκυψαν οι τρείς νέοι αλγόριθμοι που προτείνονται στα πλαίσια της διπλωματικής μας εργασίας.

2.1 Μοντέλο δικτύου και μοντέλο κόμβων

Θεωρούμε ένα σύνολο από κόμβους - αισθητήρες (πηγές, sources), κόμβους επεξεργασίας (αναμεταδότες, relays) και κόμβους – συλλέκτες (καταναλωτές, consumers) οι οποίοι συνδέονται με μονοπάτια που οδηγούν από τις πηγές στους καταναλωτές μέσω των κόμβων επεξεργασίας. Οι κόμβοι αυτοί είναι ηλεκτροκίνητοι. Οι κόμβοι - πηγές ανιχνεύουν περιβαλλοντικές παραμέτρους και σχηματίζουν με αυτές διανύσματα πλαισίου (context vectors ή CVs), τα οποία προωθούν στους καταναλωτές για περαιτέρω επεξεργασία. Απαραίτητη προϋπόθεση για την ανταλλαγή CVs μεταξύ δύο κόμβων είναι να ανήκουν στην ίδια εμβέλεια επικοινωνίας. Καθώς η εμβέλεια επικοινωνίας ενός κόμβου είναι περιορισμένη, οι κόμβοι, οι οποίοι δεν ανήκουν στην εμβέλεια επικοινωνίας του κόμβου συλλέκτη, αναμεταδίδουν της μετρήσεις τους μέσω των κόμβων αναμετάδοσης. Όλοι οι κόμβοι μπορούν να ανιχνεύουν δεδομένα, να επεξεργάζονται δεδομένα, ή και τα δύο. Ένας κόμβος επεξεργασίας διατηρεί αποθέματα μνήμης και πόρους επικοινωνίας για κάθε ρεύμα δεδομένων (λήψη και μετάδοση των δεδομένων). Ένας κόμβος - πηγή διατηρεί πόρους για την ανίχνευση και την προώθηση των δεδομένων (μετάδοση δεδομένων).



Εικόνα 2.1 Οι ροές πλαισίου μεταξύ των κόμβων.

Σε ένα δίκτυο, όπως αυτό που περιγράφεται παραπάνω, ο μηχανισμός PC3 κατανέμεται ανάμεσα σε έναν κόμβο – αποστολέα (sender node) και σε έναν κόμβο – παραλήπτη (receiver node) στους οποίους θα αναφερόμαστε ως κόμβο i και κόμβο j αντίστοιχα. Ο κόμβος i κάνει συμπίεση πλαισίου και προωθεί το συμπιεσμένο CV στον

κόμβο j. Ο κόμβος j κάνει αποσυμπίεση πλαισίου ώστε να αναπαράγει το αρχικό context vector. Έχοντας στη διάθεση του τα αρχικά δεδομένα, ο j μπορεί να συμπεριφερθεί σαν αποστολέας για άλλους κόμβους. Σε περίπτωση που το δίκτυο μας έχει δενδροειδή μορφή, ο κόμβος αποστολέας μπορεί, πρώτα, να συλλέγει τις μετρήσεις από όλα τα ρεύματα που καταλήγουν σε αυτόν και έπειτα, να εφαρμόζει τον PC3. Στην εικόνα 1, ο κόμβος i (sender) συλλέγει δεδομένα από K πηγές και κατασκευάζει με βάση αυτά ένα context vector επάνω στο οποίο θα εφαρμόσει τον PC3. Για την συμπίεση των δεδομένων ο PC3 χρησιμοποιεί την μέθοδο της ανάλυσης κύριων συνιστωσών (principal component analysis ή PCA) [5], η οποία περιγράφεται στην επόμενη ενότητα.

2.2 Ανάλυση Κύριων Συνιστωσών (Principal Component Analysis, PCA)

Η ανάλυση κύριων συνιστωσών είναι μια τεχνική η οποία δέχεται σαν είσοδο ένα σύνολο διανυσμάτων - δεδομένων, κατασκευάζει έναν χαμηλότερων διαστάσεων χώρο, ο οποίος αναπαρίσταται από ένα σύνολο ορθογώνιων διανυσμάτων, και προβάλλει τα αρχικά διανύσματα - δεδομένα στο χώρο αυτό.

Έστω ένα σύνολο από διανύσματα εισόδου $\vec{x}_i$, i=1….N, όπου N είναι το πλήθος των διανυσμάτων εισόδου. Αρχικά, υπολογίζουμε τη μέση τιμή των διανυσμάτων εισόδου, σύμφωνα με τον τύπο

$$\vec{m}_x = \frac{1}{N}\sum_{i=1}^{N}\vec{xi} \qquad (2.1)$$

Στη συνέχεια, κατασκευάζουμε τον πίνακα συνδιακύμανσης ο οποίος δίνεται από τη σχέση

$$C = \frac{1}{N}\sum_{i=1}^{N}(\vec{xi} - \vec{mx})(\vec{xi} - \vec{mx})^T \qquad (2.2)$$

Έπειτα, λύνουμε την εξίσωση

$$C\,U = U\,\Lambda \qquad (2.3)$$

και παίρνουμε σαν αποτέλεσμα τον πίνακα U ο οποίος έχει για κολώνες τα ιδιοδιανύσματα που αντιστοιχούν στον πίνακα C και τον πίνακα Λ ο οποίος είναι ο διαγώνιος πίνακας των ιδιοτιμών που αντιστοιχούν στα ιδιοδιανύσματα του U.

Τα ιδιοδιανύσματα διαμορφώνουν τη βάση ενός χώρου χαμηλότερων διαστάσεων στον οποίο προβάλλουμε τα διανύσματα εισόδου. Για ένα διάνυσμα εισόδου $\vec{y}$, η προβολή του δίνεται από τον τύπο

$$\vec{w} = U^T(\vec{y} - \vec{mx}) \qquad (2.4)$$

Το διάνυσμα $\vec{w}$ διατηρεί τους συντελεστές που επιτρέπουν την ανακατασκευή του αρχικού διανύσματος.

$$\vec{y}' = U\vec{w} + \vec{mx} = \sum_{i=1}^{N}\vec{wj}\vec{uj} + \vec{mx}, \qquad (2.5)$$

όπου $\vec{uj}$ η κολώνα του πίνακα j. Ωστόσο, τα ιδιοδιανύσματα δεν μπορούν να αναπαραστήσουν πλήρως το αρχικό διάνυσμα. Για αυτό το λόγο το ανακατασκευασμένο διάνυσμα διαφέρει από το αρχικό κατά κάποιο υπολειπόμενο διάνυσμα (residual vector) $\vec{h}$

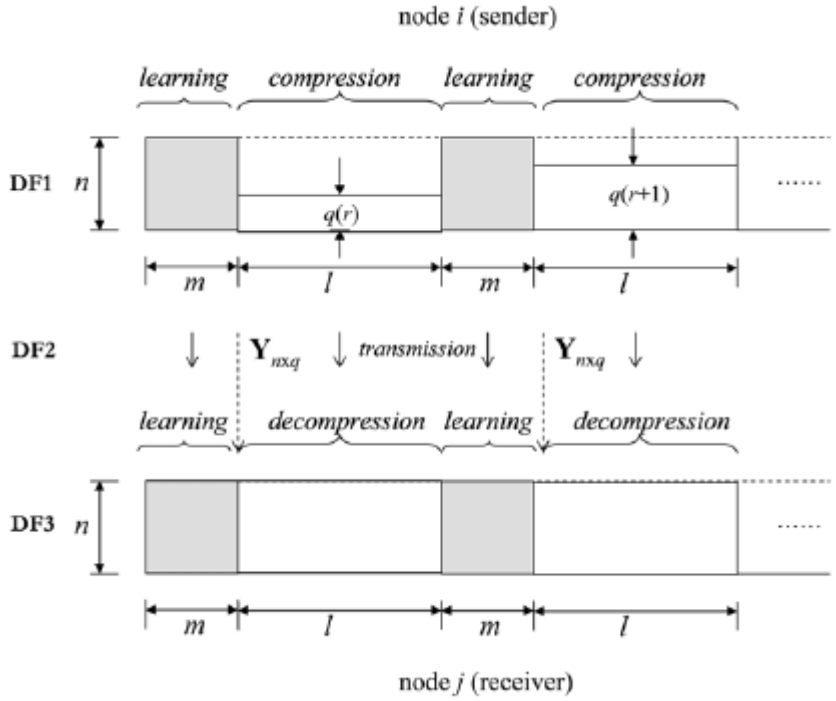
$$\vec{h} = \vec{y} - \vec{y}' \qquad (2.6)$$

Το υπολειπόμενο διάνυσμα είναι κάθετο σε όλα τα ιδιοδιανύσματα του πίνακα U.

Η βασική ιδιότητα της μεθόδου ανάλυσης κύριων συνιστωσών είναι ότι τα ιδιοδιανύσματα που αντιστοιχούν σε υψηλότερες ιδιοτιμές αποθηκεύουν υψηλότερου επιπέδου πληροφορία (οι ιδιοτιμές προσδιορίζουν το επίπεδο της διακύμανσης στην κατεύθυνση των αντίστοιχων ιδιοδιανυσμάτων). Επομένως, μπορούμε να απορρίψουμε τις κολώνες του U που αντιστοιχούν στις χαμηλότερες ιδιοτιμές, χάνοντας μικρό ποσοστό πληροφορίας και κερδίζοντας σε συμπίεση.

2.3 Το μοντέλο PC3

Η βασική ιδέα του μοντέλου PC3 είναι ότι ο κόμβος i αποφασίζει, κάθε χρονική στιγμή t, αν θα προωθήσει στον κόμβο ολόκληρο το διάνυσμα πλαισίου **x**(t) που έλαβε ή τις τιμές των συνιστωσών του context vector που αντιστοιχούν στις κύριες συνιστώσες (principal components, PCs). Οι κύριες συνιστώσες υπολογίζονται με τη μέθοδο της ανάλυσης των κύριων συνιστωσών. Πιο συγκεκριμένα, ο PC3 αποτελείται από δύο κύριες φάσεις, τη φάση της εκμάθησης (learning phase) και τη φάση της (από)συμπίεσης ((de)compression phase).



Εικόνα 2.2 Οι φάσεις εκμάθησης και (απο)συμπίεσης του PC3

Φάση εκμάθησης: Ο κόμβος i συλλέγει τα m πιο πρόσφατα context vectors **x**(t), 1<=t<=m. Κάθε context vector είναι ένα n – διάστατο διάνυσμα και κάθε μια από τις n συνιστώσες του αντιστοιχεί στην τιμή μιας από τις περιβαλλοντικές παραμέτρους που ανιχνεύει ο κόμβος i (π.χ., θερμοκρασία, υγρασία, ταχύτητα του ανέμου κ.α.). Ο κόμβος i στέλνει κάθε ένα από τα m CVs που λαμβάνει στον κόμβο j. Στο τέλος της φάσης εκμάθησης, ο κόμβος i εφαρμόζει στον πίνακα που σχηματίζεται από τα m πιο πρόσφατα CVs τη μέθοδο PCA και υπολογίζει τις κύριες συνιστώσες που αντιστοιχούν σε αυτόν. Από τις συνιστώσες ο αλγόριθμος επιλέγει τις q που αντιστοιχούν στις q μεγαλύτερες ιδιοτιμές. Οι συνιστώσες που θα επιλεγούν, σχηματίζουν έναν πίνακα $\mathbf{Y}_{nxq}$, ο οποίος θα χρησιμοποιηθεί στην φάση συμπίεσης ως βάση για τη συμπίεση των δεδομένων. Το πλήθος, q, των κύριων συνιστωσών δίνεται από τον τύπο

$$\mathbf{q = \min\{q^* | \sum_{k=1}^{q^*} \frac{eigValk}{\sum_{u=1}^{n} eigValu} \geq accuracy\}}, \quad \mathbf{(2.7)}$$

όπου $eigVal_i$ είναι την i-οστή ιδιοτιμή που αντιστοιχεί στο i-οστό ιδιοδιάνυσμα του πίνακα ιδιοδιανυσμάτων, και εξαρτάται από μια παράμετρο ακρίβειας (accuracy). Η παράμετρος ακρίβειας καθορίζει το ποσοστό της πιστότητας. Για παράδειγμα, τιμή ακρίβειας 0.9 σημαίνει ότι θέλουμε η συμπίεση των δεδομένων να μην οδηγεί σε απώλεια πληροφορίας μεγαλύτερη του 10%.

Φάση (από)συμπίεσης: Στην αρχή της φάσης συμπίεσης, κόμβος i στέλνει στον παραλήπτη τον πίνακα των κύριων συνιστωσών, Y_{nxq}, μόνο μια φορά. Σε αυτή τη φάση ο «εκπαιδευμένος» κόμβος i στέλνει στον κόμβο j μόνο τις q κύριες συνιστώσες καθενός από τα CVs που λαμβάνει, δηλαδή στέλνει τα CVs συμπιεσμένα. Η φάση (από)συμπίεσης έχει διάρκεια l χρονικές μονάδες. Επομένως, κάθε χρονική στιγμή t στο διάστημα [m, m+l] ο κόμβος i στέλνει στον κόμβο j το συμπιεσμένο CV

$$\mathbf{x_q(t) = Y^T x(t),} \quad \mathbf{(2.8)}$$

όπου $\mathbf{x}_q(t)$ είναι η προβολή του $\mathbf{x}(t)$ στη βάση $\mathbf{Y}$. Ο κόμβος j έχει την απαιτούμενη πληροφορία, δηλαδή τη βάση $\mathbf{Y}$, ώστε να αναπαράγει το αρχικό CV που αντιστοιχεί σε κάθε ένα από τα συμπιεσμένα διανύσματα που λαμβάνει στο διάστημα [m, m+l], χρησιμοποιώντας τη σχέση

$$\mathbf{\tilde{x}(t) = Y\, x_q^T(t)} \quad \mathbf{(2.9)}$$

Αξίζει να σημειωθεί ότι η αποσυμπίεση των CVs στο διάστημα [m, m+l] βασίζεται στον υπολογισμό των κύριων συνιστωσών που αντιστοιχεί στην πρόσφατη ιστορία. Αυτό σημαίνει ότι για καθορισμένη περίοδο μήκους l υποθέτουμε ότι ο αριθμός των principal components και η σειρά των αντίστοιχων ιδιοτιμών δεν αλλάζει. Η παραπάνω υπόθεση δεν ισχύει πάντα. Επομένως, ο κόμβος j αναπαράγει, την περίοδο l, τα CVs συμπεριλαμβανομένου του κόστους ανακατασκευής το οποίο δίνεται από τον τύπο

$$\mathbf{e^R = ||\tilde{x}(t) - x(t)||} \quad \mathbf{(2.10)}$$

3. ΘΕΩΡΙΑ ΒΕΛΤΙΣΤΗΣ ΠΑΥΣΗΣ (OPTIMAL STOPPING THEORY – OST)

Η Θεωρία Βέλτιστης Παύσης (Optimal Stopping Theory) μελετάει το πρόβλημα της επιλογής της χρονικής στιγμής εκτέλεσης μιας συγκεκριμένης ενέργειας, βασισμένη σε μια ακολουθία παρατηρούμενων τυχαίων μεταβλητών, προκειμένου να μεγιστοποιηθεί κάποια αναμενόμενη ανταμοιβή ή να ελαχιστοποιηθεί κάποιο αναμενόμενο κόστος. Προβλήματα αυτού του είδους συναντώνται στον τομέα της στατιστικής, όπου η εκτελούμενη ενέργεια μπορεί να αντιστοιχεί στον έλεγχο μιας υπόθεσης ή στην εκτίμηση μιας παραμέτρου, και στην περιοχή της επιχειρησιακής έρευνας, όπου η ενέργεια μπορεί να είναι η αντικατάσταση μιας μηχανής, η πρόσληψη μιας γραμματέας, η αγορά μετοχών, κ.λπ. [4].

Ιστορικά, το πρόβλημα προέκυψε στην ακολουθιακή ανάλυση στατιστικών παρατηρήσεων με τη θεωρία του κριτηρίου λόγου ακολουθιακής πιθανότητας (Wald, 1945) [9]. Η γενίκευση της ακολουθιακής ανάλυσης σε προβλήματα παύσης χωρίς στατιστική δομή έγινε από τον Snell το 1952 [8]. Στη δεκαετία του 60, η έρευνα για το πρόβλημα αυτό γνώρισε μεγάλη ώθηση [3].

3.1 Ορισμός του προβλήματος

Τα προβλήματα βέλτιστης παύσης καθορίζονται από δύο αντικείμενα [4],

(i) μια ακολουθία τυχαίων μεταβλητών $X_1, X_2, \ldots$, της οποίας η κοινή κατανομή (joint distribution) θεωρείται γνωστή, και
(ii) μια ακολουθία από συναρτήσεις απολαβής, οι οποίες λαμβάνουν πραγματικές τιμές,

$$y_0, y_1(x_1), y_2(x_1, x_2), \ldots, y_\infty(x_1, x_2, \ldots)$$

Με δεδομένα τα δύο αυτά αντικείμενα, το αντίστοιχο πρόβλημα βέλτιστης παύσης διατυπώνεται ως εξής: Παρατηρούμε την ακολουθία $X_1, X_2, \ldots$ για όσο χρόνο θέλουμε. Για κάθε $n = 1, 2, \ldots$, μετά τις παρατηρήσεις $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, μπορούμε να σταματήσουμε και να λάβουμε την προκαθορισμένη ανταμοιβή $y_n(x_1, \ldots, x_n)$, ή να συνεχίσουμε με την παρατήρηση της X_{n+1}. Αν επιλέξουμε να μην κάνουμε παρατηρήσεις, λαμβάνουμε ως ανταμοιβή τη σταθερή ποσότητα y_0. Αν δεν σταματήσουμε ποτέ τις παρατηρήσεις, λαμβάνουμε την $y_\infty(x_1, x_2, \ldots)$.

Το πρόβλημα είναι να επιλέξουμε μια χρονική στιγμή για να σταματήσουμε έτσι ώστε να βελτιστοποιήσουμε την αναμενόμενη ανταμοιβή. Για να το επιτύχουμε αυτό, μπορούμε να κάνουμε χρήση τυχαίων αποφάσεων, δηλαδή όταν φτάσουμε στο στάδιο n, έχοντας πραγματοποιήσει τις παρατηρήσεις $X_1 = x_1, \ldots, X_n = x_n$, μπορούμε να επιλέξουμε μια πιθανότητα παύσης που να εξαρτάται από τις παρατηρήσεις αυτές. Συμβολίζουμε την πιθανότητα αυτή με $\phi_n(x_1, \ldots, x_n)$. Ένας κανόνας παύσης τυχαίας κατανομής (randomized) αποτελείται από την ακολουθία αυτών των συναρτήσεων

$$\boldsymbol{\phi} = \left(\phi_0, \phi_1(x_1), \phi_2(x_1, x_2), \ldots\right) \tag{3.1}$$

όπου $0 \le \phi_n\left(x_1, \ldots, x_n\right) \le 1$ για όλα τα n και $x_1, \ldots, x_n$. Ο κανόνας παύσης δεν ακολουθεί τυχαία κατανομή (non-randomized) αν κάθε συνάρτηση $\phi_n\left(x_1, \ldots, x_n\right)$ είναι 0 ή 1.

Έτσι, η συνάρτηση ϕ_0 αντιπροσωπεύει την πιθανότητα να μην κάνουμε παρατηρήσεις, ενώ η $\phi_1\left(x_1\right)$ να σταματήσουμε μετά την πρώτη παρατήρηση κ.λπ.. Ο κανόνας παύσης $\boldsymbol{\phi}$ και η ακολουθία των παρατηρήσεων $\boldsymbol{X} = \boldsymbol{x} = \left(x_1, x_2, \ldots\right)$ καθορίζουν την τυχαία χρονική στιγμή N στην οποία γίνεται η παύση, όπου $0 \le N \le \infty$. Στην περίπτωση που η παύση δεν γίνει ποτέ, $N = \infty$. Η αθροιστική συνάρτηση κατανομής του N για $\boldsymbol{X} = \boldsymbol{x} = \left(x_1, x_2, \ldots\right)$ συμβολίζεται με $\boldsymbol{\psi} = \left(\psi_0, \psi_1, \psi_2, \ldots, \psi_n\right)$, όπου

$$\begin{aligned} \psi_n\left(x_1, \ldots, x_n\right) &= P\left(N = n \mid \boldsymbol{X} = \boldsymbol{x}\right), \qquad n = 0, 1, 2, \ldots \\ \psi_\infty\left(x_1, x_2, \ldots\right) &= P\left(N = \infty \mid \boldsymbol{X} = \boldsymbol{x}\right) \end{aligned} \tag{3.2}$$

Η αθροιστική συνάρτηση κατανομής $\boldsymbol{\psi}$ και ο κανόνας παύσης $\boldsymbol{\phi}$ συνδέονται ως εξής:

$$\begin{aligned} \psi_0 &= \phi_0 \\ \psi_1(x_1) &= \left(1 - \phi_0\right)\phi_1(x_1) \\ &\vdots \\ \psi_n(x_1, \ldots, x_n) &= \left[\left(1 - \phi_0\right)\prod_{j=1}^{n-1}\left(1 - \phi_j\left(x_1, \ldots, x_j\right)\right)\right]\phi_n\left(x_1, \ldots, x_n\right) \\ &\vdots \\ \psi_\infty(x_1, x_2, \ldots) &= 1 - \sum_{j=0}^{\infty}\psi_j\left(x_1, \ldots, x_j\right) \end{aligned} \tag{3.3}$$

όπου $\psi_\infty(x_1, x_2, \ldots)$ η πιθανότητα η παύση να μην γίνει ποτέ, με δεδομένες όλες τις παρατηρήσεις.

Το πρόβλημα λοιπόν είναι η επιλογή ενός κανόνα παύσης $\boldsymbol{\phi}$ έτσι ώστε να μεγιστοποιείται η μέση αναμενόμενη ανταμοιβή $V\left(\boldsymbol{\phi}\right)$, η οποία ορίζεται ως

$$\begin{aligned} V\left(\boldsymbol{\phi}\right) &= E\left[y_N\left(X_1, \ldots, X_N\right)\right] \\ &= E\left[\sum_{j=0}^{=\infty}\psi_j\left(X_1, \ldots, X_j\right) y_j\left(X_1, \ldots, X_j\right)\right] \end{aligned} \tag{3.4}$$

όπου ο συμβολισμός "$=\infty$" δηλώνει ότι η άθροιση είναι για τις τιμές του j από 0 μέχρι ∞, συμπεριλαμβανομένου του ∞. Ο κανόνας παύσης $\boldsymbol{\phi}$ μπορεί να εκφραστεί σε σχέση με τον τυχαίο χρόνο παύσης N ως εξής:

$$\phi_n\left(X_1,\ldots,X_n\right)=P\left(N=n\,|\,N\geq n,\boldsymbol{X}=\boldsymbol{x}\right),\qquad n=0,1,2,\ldots \tag{3.5}$$

Παρατηρήσεις

- Συχνά, η δομή του προβλήματος υπαγορεύει να θεωρήσουμε απώλεια ή κόστος αντί για ανταμοιβή. Στην περίπτωση αυτή, η συνάρτηση y_n δηλώνει το αναμενόμενο κόστος σταματώντας τη χρονική στιγμή n, και το πρόβλημα μετατρέπεται στην επιλογή ενός κανόνα παύσης για την ελαχιστοποίηση της $V\left(\boldsymbol{\phi}\right)$.
- Σε μερικές εφαρμογές, η ακολουθία ανταμοιβής περιγράφεται πιο ρεαλιστικά ως μια ακολουθία τυχαίων μεταβλητών $Y_0,Y_1,\ldots,Y_\infty$, της οποίας η κοινή συνάρτηση κατανομής με τις παρατηρήσεις $X_1,X_2,\ldots$ είναι γνωστή. Έτσι, η πραγματική τιμή της Y_n δεν είναι επακριβώς καθορισμένη τη χρονική στιγμή n όταν λαμβάνεται η απόφαση για τη συνέχιση ή την παύση των παρατηρήσεων.

3.2 Προβλήματα πεπερασμένου ορίζοντα

Ένα πρόβλημα κανόνα παύσης έχει πεπερασμένο ορίζοντα αν υπάρχει ένα γνωστό άνω όριο στον αριθμό των σταδίων που μπορεί κανείς να σταματήσει. Λέμε ότι ένα πρόβλημα έχει ορίζοντα T αν η παύση είναι υποχρεωτική μετά από τις παρατηρήσεις $X_1,\ldots,X_T$. Ένα πρόβλημα πεπερασμένου ορίζοντα μπορεί να θεωρηθεί ως ειδική περίπτωση του γενικού προβλήματος βέλτιστης παύσης θέτοντας $y_{T+1}=\cdots=y_\infty=-\infty$ Η πηγή που καθορίστηκε δεν είναι έγκυρη.

Γενικά, ένα τέτοιο πρόβλημα μπορεί να επιλυθεί με τη μέθοδο της ανάδρομης επαγωγής (backward induction). Δεδομένου ότι πρέπει να σταματήσουμε στο στάδιο T, βρίσκουμε αρχικά το βέλτιστο κανόνα στο στάδιο $T-1$. Στη συνέχεια, γνωρίζοντας το βέλτιστο κανόνα στο στάδιο $T-1$, υπολογίζουμε το βέλτιστο κανόνα στο στάδιο $T-2$, και συνεχίζουμε έτσι μέχρι το αρχικό στάδιο (στάδιο 0). Ορίζουμε την ποσότητα $V_T^{(T)}\left(x_1,\ldots,x_T\right)=y_T\left(x_1,\ldots,x_T\right)$. Στην συνέχεια, με επαγωγή από την τιμή $j=T-1$ μέχρι την τιμή $j=0$ λαμβάνουμε:

$$V_j^{(T)}\left(x_1,\ldots,x_j\right)=\max\left\{y_j\left(x_1,\ldots,x_j\right),E\left[V_{j+1}^{(T)}\left(x_1,\ldots,x_j,X_{j+1}\right)|\,X_1=x_1,\ldots,X_j=x_j\right]\right\} \tag{3.6}$$

Επαγωγικά, η ποσότητα $V_j^{(T)}\left(x_1,\ldots,x_j\right)$ αντιπροσωπεύει τη μέγιστη απολαβή που μπορεί να αποκομίσει κανείς ξεκινώντας από το στάδιο j και έχοντας κάνει τις παρατηρήσεις $X_1=x_1,\ldots,X_j=x_j$. Στο στάδιο j, συγκρίνουμε την απολαβή για παύση, $y_j\left(x_1,\ldots,x_j\right)$, με την απολαβή που αναμένουμε να λάβουμε συνεχίζοντας και

χρησιμοποιώντας το βέλτιστο κανόνα για τα στάδια $j+1, j+2, \ldots, T$, η οποία είναι ίση με $E\left[V_{j+1}^{(T)}\left(x_1,\ldots,x_j,X_{j+1}\right) \mid X_1 = x_1,\ldots,X_j = x_j\right]$. Επομένως, η βέλτιστη ανταμοιβή μας είναι η μεγαλύτερη από τις δύο αυτές ποσότητες, και είναι βέλτιστο να σταματήσουμε στο στάδιο j αν ισχύει $V_j^{(T)}\left(x_1,\ldots,x_j\right) = y_j\left(x_1,\ldots,x_j\right)$, ενώ συμφέρει να συνεχίσουμε στην αντίθετη περίπτωση. Τότε, η αξία του προβλήματος κανόνα παύσης θα είναι $V_0^{(T)}$.

Υπάρχουν διάφορα προβλήματα που μπορούν να επιλυθούν αποτελεσματικά με τη χρήση της μεθόδου ανάδρομης επαγωγής. Το πιο διάσημο από αυτά είναι το πρόβλημα της γραμματέως. Άλλα προβλήματα πεπερασμένου ορίζοντα είναι το πρόβλημα Cayley-Moser και το πρόβλημα παρκαρίσματος των MacQueen-Miller.

Εκτός από τα προβλήματα πεπερασμένου ορίζοντα, μέσω της μεθόδου ανάδρομης επαγωγής επιλύονται και τα προβλήματα μη πεπερασμένου ορίζοντα. Στην κατηγορία αυτή ανήκουν και τα προβλήματα της πώλησης αντικειμένων και του χρονικά μειωμένου συσσωρευτικού κέρδους. Σε δύο από τα προτεινόμενα μοντέλα της διπλωματικής μας, συγκεκριμένα στον infHorPC31 και στον infHorPC32, χρησιμοποιήσαμε τα παραπάνω προβλήματα, αφού τα προσαρμόσαμε στις ανάγκες μας, για να προσδιορίσουμε τον κανόνα βέλτιστης παύσης της φάσης συμπίεσης. Τα προβλήματα της πώλησης αντικειμένων και του χρονικά μειωμένου συσσωρευτικού κέρδους περιγράφονται αναλυτικά στα κεφάλαια 4 και 5.

4. Ο ΑΛΓΟΡΙΘΜΟΣ INFHORPC31

Μια από τις παραλλαγές του αλγορίθμου PC3 [1], η οποία προτείνεται στην εργασία αυτή, είναι ο αλγόριθμος infHorPC31. Ο νέος αυτός αλγόριθμος, όμοια με τον PC3, στο ξεκίνημά του έχει μια φάση εκμάθησης που διαρκεί m χρονικές μονάδες. Υποθέτουμε ότι κάθε χρονική στιγμή αναφέρεται στην λήψη ενός διανύσματος δεδομένων. Κατά τη φάση αυτή, ο πομπός στέλνει στο δέκτη το σύνολο των δειγμάτων που συγκεντρώνει μέσω των μετρήσεων των αισθητήρων, χωρίς να πραγματοποιήσει συμπίεση. Επιπλέον, μετά τη συγκέντρωση των m πρώτων δειγμάτων, υπολογίζει έναν πίνακα - βάση, μέσω της ανάλυσης κύριων συνιστωσών, που θα χρησιμεύσει στην πραγματοποίηση της συμπίεσης και θα σταλεί από τον πομπό στον δέκτη κατά το πρώτο βήμα της φάσης συμπίεσης. Η κύρια διαφορά του με τον PC3 είναι ότι, εδώ, η φάση συμπίεσης δε διαρκεί συγκεκριμένες χρονικές μονάδες (αριθμό λήψεων), αντιθέτως, ο ορίζοντας συμπίεσης είναι μη πεπερασμένος και θεωρητικά μπορεί να έχει άπειρο μήκος. Για τη λήψη της απόφασης βέλτιστης παύσης της συμπίεσης, χρησιμοποιήσαμε τη θεωρία βέλτιστης παύσης, όπως αυτή εφαρμόστηκε στο πρόβλημα της πώλησης ενός αντικειμένου μη πεπερασμένου ορίζοντα [4].

4.1 Το πρόβλημα της πώλησης ενός αντικειμένου

Έστω ότι έχουμε το πρόβλημα της πώλησης ενός αντικειμένου μη πεπερασμένου ορίζοντα, με ή χωρίς ανάκληση, που ορίζεται από ένα σύνολο τιμών και κερδών ως ακολούθως: Οι παρατηρήσεις X_1, X_2, …, X_n, που αντιπροσωπεύουν την τιμή του αντικειμένου κάθε χρονική στιγμή, υποθέτουμε ότι είναι ανεξάρτητες και ομοιόμορφα κατανεμημένες, και ακολουθούν γνωστή κατανομή F(x). Η ακολουθία ανταμοιβών (κερδών) εξαρτάται από το εάν επιτρέπεται ανάκληση των προηγούμενων παρατηρήσεων. Αν δεν επιτρέπεται ανάκληση, τότε

$$Y_0=-\infty,\ Y_1 = X_1 - c,\ \ldots,\ Y_n = X_n - n*c,\ \ldots,\ Y_\infty=-\infty,$$

ενώ, όταν επιτρέπεται ανάκληση

$$Y_0=-\infty,\ Y_1 = M_1 - c,\ \ldots,\ Y_n = M_n - n*c,\ \ldots,\ Y_\infty=-\infty,$$

όπου $c > 0$ είναι το κόστος ανά παρατήρηση, και $M_n=\max\{X_1,\ldots,X_n\}$. Αυθαίρετα, τίθεται $Y_0=-\infty$, ώστε να αναγκαστεί κανείς να λάβει τουλάχιστον μια παρατήρηση. Η ανάθεση $Y_0=-\infty$ είναι λογική, εφόσον το κόστος ενός άπειρου αριθμού παρατηρήσεων είναι άπειρο.

Έστω ότι παίρνουμε το πρόβλημα της πώλησης ενός αντικειμένου χωρίς ανάκληση:

$$\mathbf{Y_n = X_n - n*c}. \qquad \mathbf{(4.1)}$$

Θα ψάξουμε να βρούμε τον κανόνα βέλτιστης παύσης. Έστω ότι με V^* συμβολίζουμε την αναμενόμενη ανταμοιβή από έναν κανόνα βέλτιστης παύσης. Υποθέτουμε ότι πληρώνουμε c και παρατηρούμε $X_1 = x_1$. Συνεχίζοντας από το σημείο αυτό, εφόσον δεν έχουμε ανάκληση, η τιμή x_1 χάνεται και το κόστος c έχει ήδη πληρωθεί, οπότε είναι σα να ξεκινάμε το πρόβλημα ξανά από την αρχή, δηλαδή το πρόβλημα είναι αμετάβλητο στο χρόνο. Επομένως, αν συνεχίσουμε από το σημείο αυτό, μπορούμε να αποκτήσουμε ένα αναμενόμενο κέρδος V^*, αλλά τίποτε παραπάνω. Η αρχή της βελτιστότητας λέει ότι, εάν $x_1 < V^*$, συμφέρει να συνεχίσουμε, και εάν $x_1 > V^*$, θα πρέπει να σταματήσουμε τη διαδικασία. Το επιχείρημα αυτό ισχύει για κάθε στάδιο της διαδικασίας, έτσι ο κανόνας που προκύπτει από την αρχή της βελτιστότητας είναι

$$N^* = \min\{n \geq 1: X_n \geq V^*\}. \quad (4.2)$$

Το πρόβλημα τώρα είναι να υπολογιστεί το αναμενόμενο κέρδος V^*. Αυτό μπορεί να συμβεί μέσω της παρακάτω εξίσωσης βελτιστότητας

$$V^* = E(\max\{X_1, V^*\}) - c = \int_{-\infty}^{V^*} V^* dF(x) + \int_{V^*}^{\infty} x\, dF(x) - c, \quad (4.3)$$

όπου F είναι η συνάρτηση κατανομής των παρατηρήσεων X_i. Με αναδιάταξη των όρων, έχουμε

$$\int_{V^*}^{\infty} (x - V^*)\, dF(x) = c \text{ ή } E(X - V^*) = c \quad (4.4)$$

Αφού η αριστερή πλευρά είναι συνεχής στο V* και μειώνεται από το $-\infty$ έως το μηδέν, υπάρχει μία μοναδική λύση για το V* για κάθε c>0.

Υποθέτουμε ότι η F είναι ομοιόμορφη στο διάστημα (0,1).

Για $0 \leq u \leq 1$,

$$\int_u^1 (x - u) dF(x) = (1 - u)^2/2,$$

ενώ για $u < 0$,

$$\int_0^1 (x - u) dF(x) = ½ - u.$$

Εξισώνοντας με c, βρίσκουμε

$$V^* = 1 - (2c)^{1/2}, \text{ αν } c \leq 1/2, \quad (4.5)$$

$$V^* = -c + ½, \text{ αν } c > ½. \quad (4.6)$$

Ο βέλτιστος κανόνας N* υποδεικνύει να δεχτούμε την πρώτη προσφορά που θα είναι μεγαλύτερη ή ίση με V*.

Θεωρούμε τώρα το πρόβλημα με ανάκληση, $Y_0 = Y_{\infty} = -\infty$, και $Y_n = M_n - n*c$, όπου $M_n = \max\{X_1, \ldots, X_n\}$. Υποθέτουμε ότι σε κάποιο στάδιο παρατηρήσαμε $M_n = m$ και είναι βέλτιστο να συνεχίσουμε τη διαδικασία. Τότε, στο επόμενο στάδιο, εάν το M_{n+1} είναι ακόμη m, είναι ξανά βέλτιστο να συνεχίσουμε λόγω της μη μεταβλητότητας του προβλήματος στο χρόνο. Έτσι, η αρχή της βελτιστότητας δεν απαιτεί ανάκληση οποιασδήποτε προηγούμενης παρατήρησης. Επομένως, ο βέλτιστος κανόνας έχει ήδη βρεθεί και είναι ίδιος με αυτόν του προβλήματος χωρίς ανάκληση.

4.2 Εύρεση του κανόνα βέλτιστης παύσης της φάσης συμπίεσης

Για την εύρεση του κανόνα βέλτιστης παύσης της φάσης συμπίεσης, προσαρμόσαμε το πρόβλημα της πώλησης ενός αντικειμένου με σκοπό την επίτευξη μέγιστου κέρδους στο πρόβλημα της συμπίεσης δειγμάτων με σκοπό την επίτευξη ελάχιστου σφάλματος. Έτσι, έχουμε, αν y_k είναι το δείγμα που λαμβάνουμε από τους αισθητήρες, y_k' το συμπιεσμένο διάνυσμα που στέλνει ο πομπός στον δέκτη, και y_k^* το διάνυσμα που προκύπτει από την αποσυμπίεση του y_k' στον δέκτη, τότε

$$\lambda_k = \|y_k - y_k^*\| / \|y_k\|, \quad (4.7)$$

όπου ||.|| η ευκλείδεια νόρμα και $\lambda_κ \in [0,+\infty)$ ο λόγος της ποσοστιαίας διαφοράς μεταξύ του αρχικού διανύσματος και του ανακατασκευασμένου διανύσματος. Θεωρούμε ένα κατώφλι σφάλματος με τιμή στο (0,1), έστω errorThres. Αν $\lambda_κ \leq$ errorThres, τότε θεωρούμε $X_κ$ ίση με μηδέν, ενώ αν $\lambda_κ >$ errorThres, τότε $X_κ = \min\{ \lambda_κ , 1\}$.

Οι παρατηρήσεις μας στο πρόβλημα αυτό είναι $X_1, X_2, \ldots, X_n$ και δηλώνουν το σφάλμα που παρατηρείται κάθε χρονική στιγμή. Κέρδος θεωρείται η επίτευξη ελάχιστου σφάλματος. Έτσι, οδηγούμαστε, όμοια με το πρόβλημα της πώλησης ενός αντικειμένου, στην ακολουθία

$$Y_0=+\infty, Y_1 = X_1 + c, \ldots, Y_n = X_n + n*c, \ldots, Y_\infty=+\infty,$$

όπου $c > 0$ είναι το κόστος ανά παρατήρηση. Έχουμε, λοιπόν, $Y_n = X_n + n*c$. Έστω ότι με V^* συμβολίζουμε το αναμενόμενο κόστος (σφάλμα) από έναν κανόνα βέλτιστης παύσης. Σκοπός μας είναι να βρεθεί ο βέλτιστος κανόνας παύσης, ο οποίος στην περίπτωσή μας θα είναι

$$\mathbf{N^* = \min\{n \geq 1: X_n \leq V^*\}} \quad \textbf{(4.8)}$$

Το πρόβλημα τώρα είναι να υπολογιστεί το αναμενόμενο κόστος V^*. Αυτό μπορεί να συμβεί μέσω της παρακάτω εξίσωσης βελτιστότητας

$$\mathbf{V^* = E(\min\{X_1, V^*\}) + c = \int_{-\infty}^{V^*} x\, dF(x) + \int_{V^*}^{\infty} V^*\, dF(x) + c ,} \quad \textbf{(4.9)}$$

όπου F είναι η συνάρτηση κατανομής των παρατηρήσεων X_i.

Με αναδιάταξη των όρων, έχουμε

$$\int_{-\infty}^{V^*} (x - V^*)\, dF(x) = -c \quad \textbf{(4.10)}$$

Υποθέτουμε ότι η F είναι ομοιόμορφη στο διάστημα (0,1).

Για $0 \leq u \leq 1$,

$$\int_0^u (x-u) dF(x) = -u^2/2,$$

Εξισώνοντας με -c, βρίσκουμε

$$\mathbf{V^* = (2c)^{1/2}, \text{ αν } c \leq 1/2,} \quad \textbf{(4.11)}$$

Ο βέλτιστος κανόνας N* υποδεικνύει να δεχτούμε την πρώτη προσφορά που θα είναι μικρότερη ή ίση με V*.

4.3 Πλήρης περιγραφή του infHorPC31 και ποιοτική μελέτη

Έχοντας, πλέον, εντοπίσει, μέσω της αρχής βελτιστότητας, τον κανόνα για τη βέλτιστη παύση της συμπίεσης, μπορούμε να περιγράψουμε πλήρως το νέο προτεινόμενο σχήμα infHorPC31.

Ο infHorPC31 αποτελείται από τουλάχιστον μια φάση εκμάθησης και τουλάχιστον μια φάση συμπίεσης. Αρχικά, δοθέντος ενός αριθμού m βημάτων στέλνει m διανύσματα, όπως αυτά λήφθηκαν από τους αισθητήρες, από τον πομπό στον δέκτη, χωρίς να πραγματοποιεί συμπίεση. Μέσω των m αυτών διανυσμάτων με ανάλυση κύριων συνιστωσών υπολογίζει μια βάση που θα χρησιμεύσει στη συνέχεια για τη συμπίεση

των δειγμάτων. Από τη βάση αυτή κρατά μόνο τις πρώτες q κύριες συνιστώσες, σύμφωνα με μια παράμετρο ακρίβειας, την οποία, σε όλα τα πειράματα που πραγματοποιηθήκαν και θα δούμε σε επόμενο κεφάλαιο, θέτουμε πάντα ίση με 0.9. Δηλαδή, επιθυμούμε η συμπίεση να οδηγεί σε πιστότητα της τάξης του 90%. Συγκεκριμένα, η επιλογή των q κύριων συνιστωσών γίνεται, όπως και στον PC3, μέσω του τύπου

$$\mathbf{q = \min\{q^* | \sum_{k=1}^{q^*} \frac{eigValk}{\sum_{u=1}^{n} eigValu} \geq accuracy\},} \quad \textbf{(4.12)}$$

όπου με $eigVal_i$ συμβολίζουμε την i-οστή ιδιοτιμή που αντιστοιχεί στο i-οστό ιδιοδιάνυσμα του πίνακα ιδιοδιανυσμάτων (πίνακας - βάση). Στη συνέχεια, όμοια και πάλι με τον PC3, στο πρώτο βήμα της συμπίεσης στέλνεται από τον πομπό στον δέκτη η βάση που υπολογίστηκε στο m-οστό βήμα και το διάνυσμα που προκύπτει από τη συμπίεση του διανύσματος που εντοπίστηκε από τους αισθητήρες. Ο δέκτης λαμβάνει τη βάση και αποσυμπιέζει το ληφθέν διάνυσμα, παράγοντας ένα νέο ανακατασκευασμένο διάνυσμα διάστασης ίσης με το αρχικό. Τότε, υπολογίζεται το ποσοστιαίο σφάλμα της διαφοράς του μέτρου του αρχικού από το ανακατασκευασμένο διάνυσμα, λ_K. Όπως είδαμε και προηγούμενα, αν ο λόγος αυτός υπερβαίνει ένα κατώφλι, το οποίο θα τίθεται παραμετρικά, τότε θεωρούμε ως τρέχον σφάλμα είναι το ελάχιστο μεταξύ της μονάδας και της τιμής του λ_K. Αντίθετα, αν το λ_K είναι μικρότερο του κατωφλιού τότε θεωρούμε το τρέχον σφάλμα μηδενικό. Ο πομπός θα συνεχίσει να στέλνει συμπιεσμένα διανύσματα στον δέκτη, έως ότου βρεθεί ο μέγιστος χρονικός ορίζοντας συμπίεσης ο οποίος θα είναι τέτοιος ώστε το τρέχον σφάλμα θα είναι μικρότερο από το αναμενόμενο σφάλμα την επόμενη χρονική στιγμή. Όπως είναι λογικό, για κάποιο συνδυασμό τιμών των παραμέτρων του αλγορίθμου είναι πιθανό να μην χρειαστεί να υπάρξει δεύτερη φάση εκμάθησης και ως εκ τούτου δεύτερη φάση συμπίεσης. Αν, ωστόσο, για ένα συνδυασμό των τιμών των παραμέτρων διακοπεί η φάση της συμπίεσης, ενώ υπάρχουν δεδομένα που απομένουν να αποσταλούν, συνεχίζουμε σα να ξεκινούσαμε το πρόβλημα από την αρχή, κάνοντας αρχικά εκμάθηση για m βήματα, κατόπιν, υπολογίζοντας τη νέα βάση ιδιοδιανυσμάτων και τέλος, συμπιέζοντας στη συνέχεια κάθε επόμενο δείγμα, προτού αυτό αποσταλεί από τον δέκτη στον πομπό. Οι διαφορετικές φάσεις συμπίεσης μπορεί να έχουν διαφορετικό ορίζοντα, ο οποίος, μάλιστα, θα μπορούσε ανά περιπτώσεις να είναι και μικρότερος από m βήματα. Σημειώνουμε ότι σε κάθε βήμα υπολογίζεται η συνολική ενέργεια που δαπανάται, μέχρι το βήμα αυτό, για τη συμπίεση των συνολικών δεδομένων, την αποστολή αυτών, σε πακέτα με 7 bytes επικεφαλίδα και 20 bytes πληροφορία, από τον δέκτη στον πομπό, την λήψη αυτών από τον πομπό και τέλος, την αποσυμπίεση αυτών στον πομπό. Όταν υπολογίζεται η βάση προστίθεται στη συνολική ενέργεια και η ενέργεια για τον υπολογισμό της βάσης, ενώ όταν αποστέλλεται προστίθεται στη συνολική ενέργεια και η ενέργεια που καταναλώνεται για την αποστολή της σε πακέτα. Ακόμη, υπολογίζεται το μέσο σχετικό σφάλμα που προκύπτει από την ανακατασκευή του αρχικού διανύσματος.

Παρακάτω δίνεται ο αλγόριθμος σε μορφή ψευδοκώδικα.

Είσοδοι: m (ορίζοντας φάσης εκμάθησης), c (κόστος), errorThres (κατώφλι σφάλματος)

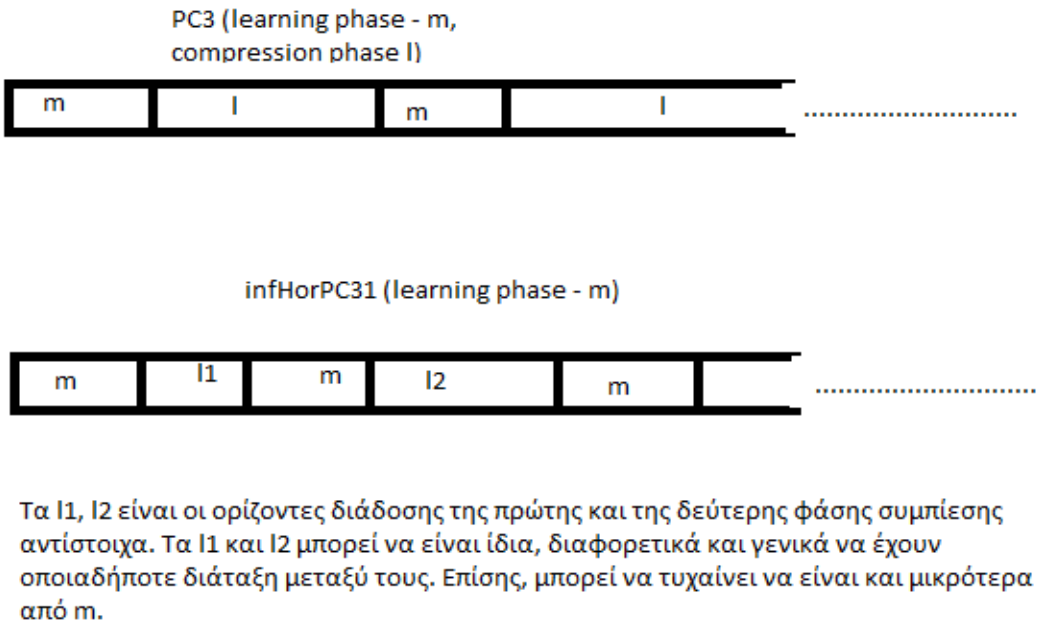
currentSample = 0

Μέχρι να τελειώσει το δείγμα

1. Στείλε m διανύσματα διάστασης n ασυμπίεστα, υπολόγισε τη βάση ιδιοδιανυσμάτων με ακρίβεια 0.9. currentSample = currentSample + m
2. Θέσε currentError = 1.1 (για να μπει τουλάχιστον μια φορά στην επανάληψη)
3. Όσο currentError > sqrt(2c) και υπάρχει δείγμα

    3.1currentSample = currentSample + 1
    3.2 Συμπίεσε το τρέχον δείγμα y και στείλε το στον δέκτη y'
    3.3 Αποσυμπίεση του δείγματος στον δέκτη, ynew
    3.4 Υπολογισμός του λόγου λκ = ||y – ynew||/||y||.
    3.5 Αν λκ <= errorThres
        currentError = 0
    3.6 Αλλιως,
        currentError = min{λκ,1}
    τέλος αν

Από τα παραπάνω συμπεραίνουμε ότι πιθανότατα, όσο μειώνουμε το κόστος και διατηρώντας σταθερό το κατώφλι σφάλματος, τόσο περισσότερη ενέργεια



Εικόνα 4.1 Οι φάσεις εκμάθησης και συμπίεσης στον πομπό των PC3 και infHorPC31 αντίστοιχα

θα κερδίζουμε – εφόσον θα γίνεται συμπίεση για περισσότερα βήματα – αλλά, και τόσο περισσότερο σφάλμα είναι πιθανό να λαμβάνουμε. Όσο μειώνουμε το κατώφλι σφάλματος, διατηρώντας σταθερό το κόστος, τόσο πιθανότερο είναι να λάβουμε λιγότερο σφάλμα και να καταναλώσουμε περισσότερη ενέργεια, εφόσον θα κάνουμε συμπίεση για λιγότερα βήματα. Βέβαια, η επίδραση της μείωσης του κατωφλιού σφάλματος εξαρτάται και από τη φύση των δεδομένων, δηλαδή αν υπάρχουν μεγάλες μεταβολές μεταξύ διαδοχικών δειγμάτων, και από το μέγεθος της τιμής της μεταβλητής του κόστους, c. Ασφαλή συμπεράσματα θα προκύψουν μόνο μετά την πειραματική μελέτη του αλγορίθμου.

Συγκριτικά με τον PC3, όταν ο ορίζοντας συμπίεσης l που επιλέγεται δεν είναι πολύ μεγάλος, μπορούμε να πούμε ότι είναι πιθανό να έχει μεγαλύτερο ενεργειακό κέρδος, εφόσον πιθανότατα θα κάνει συμπίεση για περισσότερα βήματα, αφού ακολουθεί τη θεωρία βέλτιστης παύσης για τη λήξη της συμπίεσης. Αυτό, όμως, είναι πιθανό να οδηγήσει σε μεγαλύτερη τιμή σφάλματος συγκριτικά με τον PC3. Ωστόσο, αν ο ορίζοντας συμπίεσης l στον PC3 είναι αρκετά μεγάλος συγκριτικά με τον ορίζοντα εκμάθησης m, πιθανότατα θα έχει μικρότερη κατανάλωση ενέργειας, αλλά μεγαλύτερο σφάλμα σε σχέση με τον infHorPC31. Το αν θα επιλέγαμε το σχήμα αυτό για συμπίεση έναντι του PC3 έχει να κάνει από το αν θα ενδιέφερε περισσότερο το κέρδος σε ενέργεια ή το μικρότερο σφάλμα στα δεδομένα. Διαφορετικοί συνδυασμοί στις τιμές των παραμέτρων, ασφαλώς, θα οδηγήσουν σε διαφορετικά αποτελέσματα.

5. Ο ΑΛΓΟΡΙΘΜΟΣ INFHORPC32

Ένα νέο σχήμα που προτείνεται στην εργασία αυτή για τη συμπίεση πληροφορίας πλαισίου και αποτελεί και αυτό παραλλαγή του πρωταρχικού αλγορίθμου PC3 [1] είναι ο αλγόριθμος infHorPC32. Ο infHorPC32, όμοια με τους προηγούμενους αλγορίθμους, αποτελείται από μια φάση εκμάθησης κατά την οποία ο πομπός στέλνει στον δέκτη τα m πρώτα διανύσματα που λαμβάνει από τους αισθητήρες ασυμπίεστα, υπολογίζοντας παράλληλα στο τελευταίο βήμα της φάσης εκμάθησης των πίνακα των ιδιοδιανυσμάτων που θα χρησιμοποιηθεί ως βάση για τη διαδικασία της συμπίεσης. Όμοια με τον αλγόριθμο infHorPC31, που παρουσιάστηκε στο προηγούμενο κεφάλαιο, ο infHorPC32 δεν έχει πεπερασμένο ορίζοντα συμπίεσης, αλλά επιλέγει με ένα βέλτιστο τρόπο το βήμα στο οποίο πρέπει να γίνει παύση της συμπίεσης. Ο βέλτιστος αυτός τρόπος είναι εμπνευσμένος από την εφαρμογή της αρχής της θεωρίας βέλτιστης παύσης σε προβλήματα παύσης ενός μειούμενου αθροίσματος (discounted sum), όπως το πρόβλημα του διαρρήκτη (the burglar problem) [4].

5.1 Το πρόβλημα του χρονικά μειωμένου συσσωρευτικού κέρδους

Έστω X_t η τυχαία μεταβλητή την χρονική στιγμή t. Η τιμή της μεταβλητής X_t αναφέρεται σε ποσότητα που συσσωρεύεται, π.χ., κέρδη. Η συσσώρευση S_t είναι το άθροισμα των τιμών που έχουν παρατηρηθεί από την χρονική στιγμή 1 μέχρι και t, οπότε η S_t είναι η τυχαία μεταβλητή

$$\mathbf{S_t = X_1 + X_2 + \dots + X_t} \qquad \mathbf{(5.1)}$$

Κάθε χρονική στιγμή t υπάρχει πιθανότητα 1-β να χάσουμε όλο το κέρδος μας S_t και πιθανότητα β να μην το χάσουμε, $0< \beta < 1$. Ορίζουμε την τυχαία μεταβλητή Z_t όπου

- **Z_t = 1 με πιθανότητα β, δηλαδή P{ Z_t = 1} = β, και**
- **Z_t = 0 με πιθανότητα 1-β, δηλαδή P{ Z_t = 0} = 1-β.**

(5.2)

Στόχος είναι να σταματήσουμε εκείνη τη χρονική στιγμή όπου μεγιστοποιούμε το κέρδος μας με τον φόβο ότι μπορούμε να τα χάσουμε όλα.

Ορίζουμε ως συνάρτηση κέρδους την ποσότητα $Y_t = Z_{1^*} Z_{2^*}\dots_* Z_{t^*}(X_1+ \dots+ X_t)$. Αν τη χρονική στιγμή t έχουμε $Z_t = 0$, τότε το κέρδος μας είναι $Y_t = 0$. Διαφορετικά, το κέρδος μας είναι $Y_t = (X_1+ \dots+ X_t) = S_t$. Το πρόβλημά μας είναι να βρούμε το t* έτσι ώστε

$$\mathbf{t^* = argmax\ Y_t,\ t = 1,2,\dots} \qquad \mathbf{(5.3)}$$

Δηλαδή, θέλουμε να παραμένουμε όλο και περισσότερο στην διαδικασία για να συσσωρεύσουμε όλο και περισσότερες X_t τιμές, αλλά καθώς παρατείνουμε τη διαδικασία, τόσο μεγαλύτερο είναι το κόστος, αν τα χάσουμε όλα.

Βάσει της αρχής της στοχαστικής βελτιστότητας, η αναδρομική εξίσωση είναι:

$$\mathbf{Y_t = max(S_t, E\{Y_{t+1}\})} \qquad \mathbf{(5.4)}$$

και σταματάμε τη διαδικασία X_t και συνεπώς το άθροισμα S_t όταν $Y_t > E\{Y_{t+1}\}$.

Εάν μέχρι και την χρονική στιγμή t δεν έχουμε χάσει το όλο συσσωρευμένο κέρδος, τότε έχουμε

$$Z_{1^*} Z_{2^*}\dots_* Z_t = 1 \text{ και } Y_t = S_t .$$

Για να τερματίσουμε τη διαδικασία κινδυνεύοντας με πιθανότητα 1-β να χάσουμε το όλο συσσωρευμένο κέρδος, η τιμή $E\{Y_{t+1}\}$ πρέπει να είναι μικρότερη από την Y_t, ή

$$E\{Y_{t+1}\} = E\{Z_{t+1*}(S_t + X_{t+1})\} = E\{Z_{t+1}\}_*(S_t + E\{X_{t+1}\}) =$$
$$= (P(Z_{t+1} = 1)*1 + P(Z_{t+1} = 0)*0) *(S_t + \mu) =$$
$$= \beta*(S_t + \mu),$$

όπου μ είναι η μέση τιμή του X_t την χρονική στιγμή t, δηλαδή $\mu = E\{ X_t \}$.

Τότε, πρέπει

$$Y_t = S_t > E\{ Y_{t+1} \} = \beta*(S_t + \mu),$$

πράγμα που σημαίνει ότι σταματάμε στο πρώτο t έτσι ώστε το άθροισμα S_t να είναι μεγαλύτερο από την ποσότητα $\beta*\mu/(1 – \beta)$, δηλαδή ο κανόνας βέλτιστης παύσης είναι

$$\mathbf{t^* = \min\{t: S_t > \beta*\mu / (1 – \beta)\}.} \quad \mathbf{(5.5)}$$

5.2 Εύρεση του κανόνα βέλτιστης παύσης της φάσης συμπίεσης

Για την εύρεση του κανόνα βέλτιστης παύσης της φάσης συμπίεσης, προσαρμόσαμε το πρόβλημα του χρονικά μειωμένου συσσωρευτικού κέρδους στο πρόβλημα της συμπίεσης δειγμάτων με σκοπό την επίτευξη ελάχιστου σφάλματος. Συγκεκριμένα, δοθέντος ενός κατωφλιού σφάλματος (error threshold) θ και του λόγου της ποσοστιαίας διαφοράς μεταξύ του αρχικού διανύσματος και του ανακατασκευασμένου στο δέκτη διανύσματος λκ, ορίζουμε X_t την τυχαία μεταβλητή την χρονική στιγμή t, η οποία θα λαμβάνει την τιμή 1, όταν $\lambda\kappa \leq \theta$, και τιμή 0 σε διαφορετική περίπτωση. Καταλαβαίνουμε ότι η τιμή της μεταβλητής X_t αναφέρεται σε κέρδος που συσσωρεύεται, με το κέρδος να ορίζεται ως αριθμός των δειγμάτων που έχουν ποσοστιαίο σφάλμα ανακατασκευής μικρότερο από θ.

Όμοια με το παραπάνω, ορίζουμε ως συνάρτηση κέρδους την ποσότητα

$$\mathbf{Y_t = Z_{1*} Z_{2*}….* Z_{t*}(X_1 + … + X_t) = \beta^t_* S_t,\ \beta \in (0, 1)}. \quad \mathbf{(5.6)}$$

Ακολουθώντας την παραπάνω λογική καταλήγουμε στον κανόνα βέλτιστης παύσης

$$\mathbf{t^* = \min\{t: S_t > \beta*E\{ X_t \} / (1 – \beta)\}.} \quad \mathbf{(5.7)}$$

Εδώ, όμως, δε θεωρούμε σταθερή τη μέση τιμή των δειγμάτων X_t και ίση με ½ (μέσος όρος των τιμών 0 και 1 που λαμβάνει το δείγμα), αλλά λαμβάνουμε ότι

$$\mathbf{E\{X_t\} = P(\lambda\kappa \leq \theta) * 1 + P(\lambda\kappa > \theta) * 0 = P(\lambda\kappa \leq \theta).} \quad \mathbf{(5.8)}$$

Καταλήγουμε, συνεπώς, στον κανόνα βέλτιστης παύσης

$$\mathbf{t^* = \min\{t: S_t > \beta* P(\lambda\kappa \leq \theta) / (1 – \beta)\}.} \quad \mathbf{(5.9)}$$

Οι μεταβλητές λκ, ωστόσο, δεν ακολουθούν κάποια γνωστή κατανομή. Για το λόγο αυτό, χρησιμοποιούμε τον εκτιμητή πυκνότητας Kernel (Kernel Density Estimator ή KDE) , οποίος δοθέντων κάποιων δειγμάτων εκτιμά την συνάρτηση πυκνότητας πιθανότητας (probability density function ή PDF) της κατανομής που αυτά ακολουθούν. Συγκεκριμένα, δοθέντων των δειγμάτων $(x_1, x_2, …, x_N)$ τα οποία προκύπτουν από μια άγνωστη κατανομή, ο εκτιμητής Kernel της κατανομής αυτής ορίζεται από την ακόλουθη εξίσωση:

$$\mathbf{P(x)} = \frac{1}{N} * \sum_{i=1}^{N} \frac{1 * e^{-\frac{(x-x_i)^2}{2}}}{\sqrt{2*\pi}}. \quad \textbf{(5.10)}$$

Η P(x) δηλώνει τη συνάρτηση πυκνότητας πιθανότητας. Λαμβάνουμε την συσσωρευτική συνάρτηση κατανομής (cumulative distribution function ή CDF):

$$\mathbf{CDF(x)} = \frac{1}{N} * \sum_{i=1}^{N} \frac{1}{2}\left(1 + erf\left(\frac{(x-x_i)}{\sqrt{2}}\right)\right). \quad \textbf{(5.11)}$$

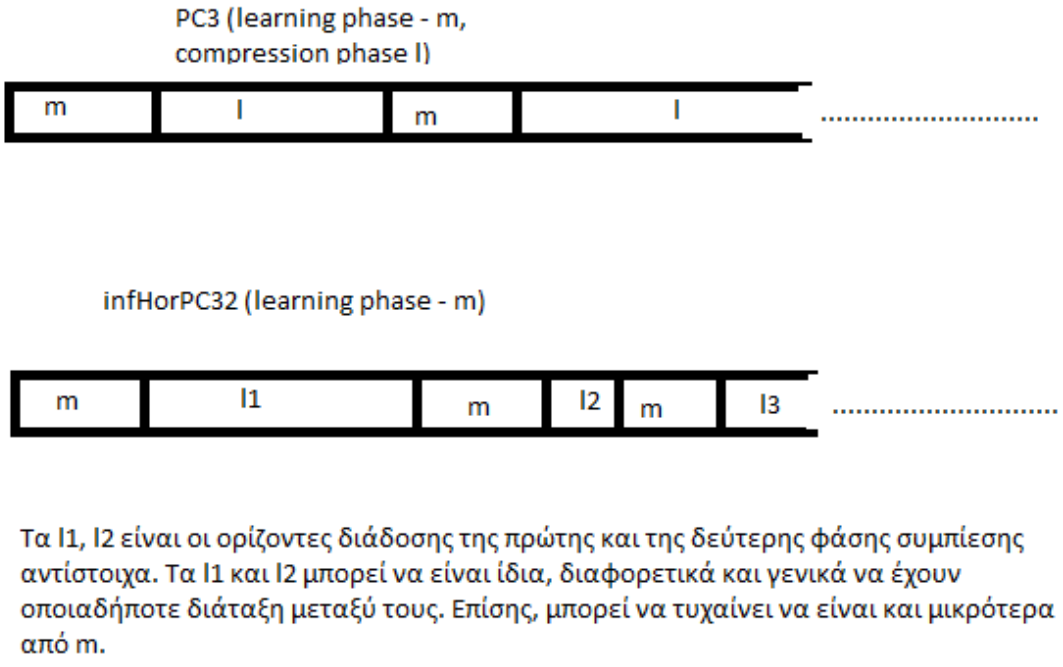
Έχουμε ότι CDF(x) = P(X ≤ x) και ότι erf(.) είναι η συνάρτηση σφάλματος (error function) της γκαουσιανής συνάρτησης. Μια προσέγγιση της συνάρτησης σφάλματος δίνεται από τον εξής τύπο:

$$\mathbf{erf(x)} \approx \sqrt{1 - e^{-x^2 \frac{\frac{4}{\pi} + a*x^2}{1 + a*x^2}}}, \quad \textbf{(5.12)}$$

όπου η παράμετρος α είναι ίση με 0.14.

5.3 Πλήρης περιγραφή του infHorPC32 και ποιοτική μελέτη

Δεδομένου, πλέον, του κανόνα βέλτιστης παύσης, είμαστε σε θέση να δώσουμε την πλήρη περιγραφή του νέου σχήματος συμπίεσης πληροφορίας πλαισίου με μη πεπερασμένο ορίζοντα συμπίεσης infHorPC32.



Εικόνα 5.1 Οι φάσεις εκμάθησης και συμπίεσης στον πομπό των PC3 και infHorPC32 αντίστοιχα

Ο infHorPC32 αποτελείται, όπως προαναφέρθηκε και όμοια με τον infHorPC31, από τουλάχιστον μια φάση εκμάθησης και τουλάχιστον μια φάση συμπίεσης. Αρχικά, δοθέντος ενός αριθμού m βημάτων στέλνει m διανύσματα, όπως αυτά λήφθηκαν από τους αισθητήρες, από τον πομπό στον δέκτη, χωρίς να πραγματοποιεί συμπίεση. Μέσω των m αυτών διανυσμάτων με ανάλυση κύριων συνιστωσών υπολογίζει μια βάση που

θα χρησιμεύσει στη συνέχεια για τη συμπίεση των δειγμάτων. Από τη βάση αυτή κρατά μόνο τις πρώτες q κύριες συνιστώσες, σύμφωνα με μια παράμετρο ακρίβειας, την οποία, σε όλα τα πειράματα που πραγματοποιηθήκαν, θέτουμε πάντα ίση με 0.9. Δηλαδή, επιθυμούμε η συμπίεση να οδηγεί σε πιστότητα της τάξης του 90%. Συγκεκριμένα, η επιλογή των q κύριων συνιστωσών γίνεται, όπως και στον PC3, μέσω του τύπου

$$\mathbf{q = \min\{q^* | \sum_{k=1}^{q^*} \frac{eigValk}{\sum_{u=1}^{n} eigValu} \geq accuracy\}}, \quad \mathbf{(5.13)}$$

όπου με $eigVal_i$ συμβολίζουμε την i-οστή ιδιοτιμή που αντιστοιχεί στο i-οστό ιδιοδιάνυσμα του πίνακα ιδιοδιανυσμάτων (πίνακας - βάση). Στη συνέχεια, όμοια με τα προηγούμενα δύο σχήματα, στο πρώτο βήμα της συμπίεσης στέλνεται από τον πομπό στον δέκτη η βάση που υπολογίστηκε στο m-οστό βήμα και το διάνυσμα που προκύπτει από τη συμπίεση του διανύσματος που εντοπίστηκε από τους αισθητήρες. Ο δέκτης λαμβάνει τη βάση και αποσυμπιέζει το ληφθέν διάνυσμα, παράγοντας ένα νέο ανακατασκευασμένο διάνυσμα διάστασης ίσης με το αρχικό. Τότε, υπολογίζεται το ποσοστιαίο σφάλμα της διαφοράς του μέτρου του αρχικού από το ανακατασκευασμένο διάνυσμα, $\lambda_κ$. Αν ο λόγος αυτός είναι μικρότερος από το δοθέν κατώφλι σφάλματος, τότε προστίθεται στο άθροισμα του συνολικού κέρδους μια μονάδα, διαφορετικά το άθροισμα παραμένει ως έχει. Σε κάθε περίπτωση υπολογίζεται η πιθανότητα $P(\lambda κ \leq \theta)$ μέσω της συσσωρευτικής συνάρτησης κατανομής, που περιγράψαμε προηγούμενα. Υπολογίζεται, επίσης, ο παράγοντας με τον οποίο συγκρίνουμε το συνολικό κέρδος για να δούμε αν θα πρέπει να λήξει η φάση της συμπίεσης. Έστω ότι τον παράγοντα αυτό τον ονομάζουμε παράγοντα τερματισμού και υπολογίζεται ως εξής:

$$\mathbf{stopFactor = (b/(1-b)) * p}, \quad \mathbf{(5.14)}$$

όπου b είναι η πιθανότητα κέρδους που δίνεται, όπως και το κατώφλι σφάλματος, παραμετρικά στον αλγόριθμο. Αν το συνολικό κέρδος είναι μικρότερο από τον παράγοντα τερματισμού, τότε η διαδικασία της συμπίεσης συνεχίζεται και για το επόμενο δείγμα. Αν, όμως, είναι μεγαλύτερος από τον παράγοντα τερματισμού, η διαδικασία συμπίεσης τερματίζεται και το σύστημα κάνει και πάλι εκμάθηση για m χρονικές μονάδες, υπολογίζοντας μια νέα βάση ιδιοδιανυσμάτων για την επόμενη περίοδο συμπίεσης. Και σε αυτό το σχήμα, όπως και στο infHorPC31, για κάποιο συνδυασμό τιμών των παραμέτρων του αλγορίθμου είναι πιθανό να μην χρειαστεί να υπάρξει δεύτερη φάση εκμάθησης και ως εκ τούτου δεύτερη φάση συμπίεσης. Τέλος, και για το σχήμα αυτό είναι λογικό οι διαφορετικές φάσεις συμπίεσης να τυχαίνει να έχουν διαφορετικό ορίζοντα, ο οποίος, μάλιστα, θα μπορούσε ανά περιπτώσεις να είναι και μικρότερος από m βήματα.

Σημειώνουμε ότι, και σε αυτό το σχήμα, σε κάθε βήμα υπολογίζεται η συνολική ενέργεια που δαπανάται, μέχρι το βήμα αυτό, για τη συμπίεση των συνολικών δεδομένων, την αποστολή αυτών, σε πακέτα με 7 bytes επικεφαλίδα και 20 bytes πληροφορία, από τον δέκτη στον πομπό, την λήψη αυτών από τον πομπό και τέλος, την αποσυμπίεση αυτών στον πομπό. Όταν υπολογίζεται η βάση προστίθεται στη συνολική ενέργεια και η ενέργεια για τον υπολογισμό της βάσης, ενώ όταν αποστέλλεται προστίθεται στη συνολική ενέργεια και η ενέργεια που καταναλώνεται για την αποστολή της σε πακέτα. Ακόμη, υπολογίζεται το μέσο σχετικό σφάλμα που προκύπτει από την ανακατασκευή του αρχικού διανύσματος.

Παρακάτω δίνεται ο αλγόριθμος σε μορφή ψευδοκώδικα.

Είσοδοι: m (ορίζοντας φάσης εκμάθησης), b (πιθανότητα κέρδους), errorThres (κατώφλι σφάλματος)

currentSample = 0

ErrorVector = []

Μέχρι να τελειώσει το δείγμα

1.Στείλε m διανύσματα διάστασης n ασυμπίεστα, υπολόγισε τη βάση ιδιοδιανυσμάτων με ακρίβεια 0.9. currentSample = currentSample + m

2.Θέσε totalGain = 0

3.Θέσε stopFactor = 1.1 (για να μπει τουλάχιστον μια φορά στην επανάληψη)

4. Όσο totalGain < stopFactor και υπάρχει δείγμα
   4.1currentSample = currentSample + 1
   4.2 Συμπίεσε το τρέχον δείγμα y και στείλε το στον δέκτη y'
   4.3 Αποσυμπίεση του δείγματος στον δέκτη, ynew
   4.4 Υπολογισμός του λόγου λκ = ||y – ynew||/||y||.
   4.6 Αν λκ <= errorThres
   totalGain = totalGain + 1
   τέλος αν
   4.7 Πρόσθεσε στο ErrorVector την τρέχουσα τιμή λκ.
   4.8 p = cdf(errorThres, ErrorVector)
   4.9 stopFactor = (b/(1-b))*p

Η πρόβλεψη της πιθανής συμπεριφοράς του συγκεκριμένου σχήματος, με αλλαγή στην τιμή μιας παραμέτρου και ταυτόχρονα διατήρηση σε σταθερή τιμή της άλλης, είναι πολύ πιο δύσκολη συγκριτικά με το σχήμα infHorPC31, εφόσον εδώ ο παράγοντας τερματισμού μεταβάλλεται με την έλευση ενός νέου διανύσματος, ενώ στον infHorPC31 ήταν πάντοτε σταθερός και ίσος με $\sqrt{(2 * c)}$. Γενικά, μπορούμε να εκτιμήσουμε ότι με αύξηση της πιθανότητας κέρδους θα έχουμε αύξηση του ορίζοντα συμπίεσης και ως εκ τούτου μικρότερη κατανάλωση ενέργειας, εφόσον η αύξηση αυτή θα σηματοδοτήσει την αύξηση του παράγοντα τερματισμού. Ωστόσο, δεν μπορούμε να κάνουμε κάποια ασφαλή πρόβλεψη για το μέσο συνολικό σχετικό σφάλμα, για την ίδια περίπτωση, από τη στιγμή που αυτό εξαρτάται και από το εάν όντως επιλέχθηκε ο σωστός χρόνος για τον τερματισμό της συμπίεση. Αν η αλλαγή στην κατανομή των δεδομένων ανιχνεύθηκε σχετικά αργά, ως αποτέλεσμα θα έχουμε συμπίεση αρκετών διανυσμάτων βάσει μιας παλιάς βάσης, η οποία δεν αντιπροσωπεύει επαρκώς τα νέα δείγματα και οδηγεί σε συμπίεση με χαμηλή πιστότητα και μεγάλο σφάλμα στην ανακατασκευή. Με μείωση του κατωφλιού σφάλματος, διατηρώντας σταθερή την τιμή της πιθανότητας κέρδους, είναι πιθανό λιγότερα δείγματα να δίνουν ποσοστιαία διαφορά του αρχικού από το ανακατασκευασμένο διάνυσμα μικρότερη του κατωφλιού σφάλματος. Έτσι, το συνολικό κέρδος θα αυξάνεται με αργό ρυθμό και θα γίνεται συμπίεση για περισσότερα δείγματα με αποτέλεσμα μεγαλύτερο ενεργειακό κέρδος. Βέβαια, αυτό εξαρτάται σημαντικά από τα ίδια τα δεδομένα, τα οποία αν διαφέρουν ούτως ή άλλως ελάχιστα μεταξύ τους θα έχουν μικρό σχετικό σφάλμα στην ανακατασκευή, με αποτέλεσμα το συνολικό κέρδος να αυξάνεται γρήγορα, μειώνοντας τοιουτοτρόπως το μήκος την περιόδου συμπίεσης

και οδηγώντας σε μεγαλύτερη κατανάλωση ενέργειας, αλλά πιθανότατα μικρότερο σφάλμα από τη στιγμή που η βάση θα ανανεώνεται συχνότερα. Η συνολική συμπεριφορά του σχήματος εξαρτάται από την επιλογή κατάλληλων τιμών στο συνδυασμό των δυο παραμέτρων. Σε κάθε περίπτωση, φαίνεται ότι το σχήμα αυτό θα δίνει μεγαλύτερο ενεργειακό κέρδος με παράλληλη αύξηση του σφάλματος, συγκριτικά με τον αρχικό μας PC3 με λογικό ορίζοντα συμπίεσης l. Σε σχέση με τον infHorPC31, είναι δύσκολο να βγουν συμπεράσματα χωρίς κάποια πειραματική μελέτη. Πιθανότατα, για την ίδια τιμή κατωφλιού σφάλματος ο infHorPC32 παίρνει καλύτερη απόφαση τερματισμού της φάσης συμπίεσης, εφόσον ο παράγοντας τερματισμού δεν είναι σταθερός σε κάθε περίπτωση, αντίθετα μεταβάλλεται με την έλευση κάθε νέου δείγματος λαμβάνοντας υπόψη το ποσοστιαίο σφάλμα ανακατασκευής στον δέκτη του νέου αυτού δείγματος μέσα από την πιθανότητα $P(\lambda\kappa \leq \theta)$ (p στον ψευδοκώδικα). Μένει, λοιπόν, να αποδειχθεί ποιος εκ των δύο θα έχει μικρότερη κατανάλωση ενέργειας και ποιος μικρότερο σφάλμα.

6. Ο ΑΛΓΟΡΙΘΜΟΣ INCPCA

Ο τελευταίος αλγόριθμος που θα παρουσιαστεί στα πλαίσια αυτής της εργασίας είναι ο incPCA. Ο incPCA περιλαμβάνει μια φάση εκμάθησης κατά την οποία ο πομπός στέλνει τα m πρώτα δείγματα που λαμβάνει από τους αισθητήρες στο δέκτη, χωρίς να τα συμπιέσει. Στο τέλος της φάσης εκμάθησης, ο πομπός υπολογίζει, μέσω της ανάλυσης κύριων συνιστωσών, τον πίνακα των ιδιοδιανυσμάτων, ο οποίος θα χρησιμοποιηθεί ως βάση για τη συμπίεση των δεδομένων. Στη συνέχεια, για κάθε νέα μέτρηση που λαμβάνει, ο πομπός προβάλλει το διάνυσμα στην υπάρχουσα βάση και το ανακατασκευάζει. Έπειτα, υπολογίζει το σχετικό σφάλμα ανάμεσα στο αρχικό διάνυσμα και το ανακατασκευασμένο. Αν το σφάλμα είναι μικρότερο από ένα κατώφλι, ο πομπός στέλνει στο δέκτη το δείγμα συμπιεσμένο και μόνο για το πρώτο βήμα μετά τη φάση της εκμάθησης στέλνει και τη βάση. Αλλιώς, ο πομπός υπολογίζει τη βάση που αντιστοιχεί στο νέο σύνολο παρατηρήσεων που προέκυψε με την προσθήκη της νέας μέτρησης που λήφθηκε, χρησιμοποιώντας τη μέθοδο της αυξητικής ανάλυσης κύριων συνιστωσών [2,5] και εφαρμόζει διάφορους ελέγχους για να αποφασίσει αν θα ανανεώσει τη βάση ή θα συνεχίσει την συμπίεση με την παλιά βάση.

6.1 Αυξητική Ανάλυση Κύριων Συνιστωσών

Η ανάλυση κύριων συνιστωσών είναι μια μαθηματική μέθοδος, η οποία για ένα σύνολο παρατηρήσεων κατασκευάζει έναν μικρότερων διαστάσεων χώρο ο οποίος αποτελείται από ένα σύνολο ορθογώνιων διανυσμάτων (κύριες συνιστώσες) και στον οποίο μπορούμε να αναπαραστήσουμε τα δεδομένα χωρίς μεγάλη απώλεια πληροφορίας, δηλαδή μας επιτρέπει να συμπιέσουμε τα δεδομένα. Για κάθε νέα παρατήρηση, το κόστος επαναϋπολογισμού των κύριων συνιστωσών, χρησιμοποιώντας την παραπάνω μέθοδο, είναι πολύ μεγάλο. Η μέθοδος αυξητικής ανάλυσης κύριων συνιστωσών μας επιτρέπει, όταν έχουμε μια νέα παρατήρηση, αντί να υπολογίσουμε εκ νέου το σύνολο των κύριων συνιστωσών να ανανεώσουμε το ήδη υπάρχον. Πιο συγκεκριμένα:

Έστω ότι έχουμε ένα σύνολο N παρατηρήσεων. Κάθε παρατήρηση είναι ένα διάνυσμα γραμμή διάστασης (1 x n). Με $\bar{x}$ συμβολίζουμε τη μέση τιμή των παρατηρήσεων, με U_{np} συμβολίζουμε τον πίνακα που έχει σαν γραμμές τα p ιδιοδιανύσματα (κύριες συνιστώσες) που προέκυψαν από την ανάλυση κύριων συνιστωσών και με Λ_{pp} τον διαγώνιο πίνακα που περιέχει τις ιδιοτιμές που αντιστοιχούν σε αυτά τα ιδιοδιανύσματα. Η προβολή, $\underline{g}$, μιας παρατήρησης, $\underline{z}$, στο χώρο που δημιουργείται από τα ιδιοδιανύσματα – κύριες συνιστώσες είναι ένα p - διάστατο διάνυσμα και ισούται με

$$\underline{g} = U_{np}^{T} (\underline{z} - \bar{x}) \quad (6.1)$$

Το διάνυσμα $\underline{g}$ μπορεί να προβληθεί ξανά πίσω στο n – διάστατο χώρο, αλλά με κάποια μικρή απώλεια πληροφορίας, η οποία αναπαρίσταται από το υπολειπόμενο διάνυσμα (residue vector) h

$$h = (\underline{z} - \bar{x}) - U_{np}\,\underline{g} \quad (6.2)$$

Έστω ότι έρχεται μια καινούρια παρατήρηση, $\underline{y}$. Θέλουμε να υπολογίσουμε τη νέα βάση – σύνολο κύριων συνιστωσών για το νέο σύνολο παρατηρήσεων, μέσω της αυξητικής ανάλυσης κύριων συνιστωσών. Η νέα μέση τιμή των παρατηρήσεων μας δίνεται από τον τύπο

$$\bar{x}' = \frac{1}{N+1}(N\bar{x} + \underline{y}) \quad (6.3)$$

Ο νέος πίνακας συνδιακύμανσης υπολογίζεται ως εξής:

$$C'_{nn} = \frac{N}{N+1} C_{nn} + \frac{N}{(N+1)^2} \underline{y}'(\underline{y}')^T \quad (6.4)$$

Το σύστημα που πρέπει να λύσουμε για να υπολογίσουμε τα νέα ιδιοδιανύσματα και τις νέες ιδιοτιμές είναι

$$C'_{nn} U'_{nn} = U'_{nn} \Lambda'_{nn} \quad (6.5)$$

Αρχικά, υποθέτουμε ότι το νέο πλήθος κύριων συνιστωσών ισούται με q=p + 1, όπου p είναι το πλήθος των ιδιονυσμάτων της παλιάς βάσης. Σε περίπτωση που η επιπρόσθετη ιδιοτιμή είναι μικρή, θα την απορρίψουμε, αργότερα, μαζί με το αντίστοιχο ιδιοδιάνυσμα. Το νέο ανανεωμένο σύνολο των ιδιοδιανυσμάτων πρέπει να είναι μια περιστροφή, $R_{(p+1)(p+1)}$, του τρέχοντος συνόλου και πρέπει να περιέχει ένα ορθογώνιο μοναδιαίο διάνυσμα. Ως επιπλέον διάνυσμα επιλέγουμε το μοναδιαίο residue διάνυσμα, το οποίο ισούται με

$$\hat{h} = \begin{cases} h/\|h\|_2, & \text{αν } \|h\|_2 \neq 0 \\ 0, & \text{αλλιώς (η νέα παρατήρηση περιέχεται στον τρέχοντα ιδιοχώρο)} \end{cases} \quad (6.6)$$

Στη συνέχεια, θέτουμε

$$U'_{nq} = [U_{np}, \hat{h}] R_{(p+1)(p+1)} \quad (6.7)$$

Η εξίσωση (6.3) με την εφαρμογή των τύπων (6.2) και (6.5) διαμορφώνεται ως εξής:

$$\left(\frac{N}{N+1} C_{nn} + \frac{N}{(N+1)^2} \underline{y}'(\underline{y}')^T\right)[U_{np}, \hat{h}] R_{(p+1)(p+1)} = [U_{np}, \hat{h}] R_{(p+1)(p+1)} \Lambda'_{(p+1)(p+1)}$$

Πολλαπλασιάζουμε από αριστερά και τα δυο μέλη της εξίσωσης με $[U_{np}, \hat{h}]^T$, οπότε έχουμε

$$[U_{np}, \hat{h}]^T \left(\frac{N}{N+1} C_{nn} + \frac{N}{(N+1)^2} \underline{y}'(\underline{y}')^T\right)[U_{np}, \hat{h}] R_{(p+1)(p+1)} = R_{(p+1)(p+1)} \Lambda'_{(p+1)(p+1)} \quad (6.8)$$

Η εξίσωση (6.8) περιγράφει ένα πρόβλημα το οποίο έχει σαν λύση ιδιοδιανύσματα $R_{(p+1)(p+1)}$ και ιδιοτιμές $\Lambda'_{(p+1)(p+1)}$. Το πρώτο μέλος της εξίσωσης περιλαμβάνει δύο όρους. Ο πρώτος όρος είναι ο

$$[U_{np}, \hat{h}]^T C_{nn} [U_{np}, \hat{h}] = \begin{pmatrix} U_{np}^T C_{nn} U_{np} & U_{np}^T C_{nn} \hat{h} \\ \hat{h}^T C_{nn} U_{np} & \hat{h}^T C_{nn} \hat{h} \end{pmatrix} \approx \begin{pmatrix} \Lambda_{pp} & \underline{0} \\ \underline{0}^T & 0 \end{pmatrix} \quad (6.9)$$

Η παραπάνω ισότητα ισχύει διότι $C_{nn} \approx U_{np} \Lambda_{pp} U_{np}^T$ και το διάνυσμα $\underline{h}$ είναι ορθογώνιο. Επίσης, το $\underline{0}$ είναι ένα p – διάστατο διάνυσμα από 0. Ο δεύτερος όρος είναι ο

$$[U_{np}, \hat{h}]^T \underline{y}'(\underline{y})^T [U_{np}, \hat{h}] = \begin{bmatrix} U_{np}^T \underline{y}'(\underline{y})^T U_{np} & U_{np}^T \underline{y}'(\underline{y})^T \hat{h} \\ \hat{h}^T \underline{y}'(\underline{y})^T U_{np} & \hat{h}^T \underline{y}'(\underline{y})^T \hat{h} \end{bmatrix} \approx \begin{bmatrix} \underline{g}\,\underline{g}^T & \gamma \underline{g} \\ \gamma \underline{g}^T & \gamma^2 \end{bmatrix} \quad (6.10)$$

Η παραπάνω ισότητα ισχύει διότι $\underline{g} = U_{np}^{\ T} (\underline{y} - \bar{x})$ και $h = (\underline{y} - \bar{x}) - U_{np}\, \underline{g}$. Επίσης, $\gamma = \hat{h}^T \underline{y}'$. Η εξίσωση (6.8), με την εφαρμογή των τύπων (6.9) και (6.10), γράφεται ως εξής:

$$\left(\frac{N}{N+1} \begin{bmatrix} \Lambda_{pp} & \underline{0} \\ \underline{0}^T & 0 \end{bmatrix} + \frac{N}{(N+1)^2} \begin{bmatrix} \underline{g}\,\underline{g}^T & \gamma \underline{g} \\ \gamma \underline{g}^T & \gamma^2 \end{bmatrix} \right) R_{(p+1)(p+1)} = R_{(p+1)(p+1)} \Lambda'_{(p+1)(p+1)}$$

Η παραπάνω εξίσωση είναι της μορφής $D_{(p+1)(p+1)} R_{(p+1)(p+1)} = R_{(p+1)(p+1)} \Lambda'_{(p+1)(p+1)}$ και η επίλυση της μας δίνει σαν αποτέλεσμα τα ζητούμενα ιδιοδιανύσματα και τις ζητούμενες ιδιοτιμές για το νέο σύνολο των N+1 παρατηρήσεων.

6.2 Πλήρης περιγραφή του incPCA και ποιοτική μελέτη

Έχοντας κάνει μια λεπτομερή περιγραφή της μεθόδου αυξητικής ανάλυσης κύριων συνιστωσών, πάνω στην οποία βασίζεται ο incPCA, μπορούμε πλέον να περιγράψουμε πλήρως το νέο αυτό αλγόριθμο.

Ο incPCA ξεκινάει, όπως και όλοι οι άλλοι αλγόριθμοι που περιγράφηκαν προηγουμένως, με μια φάση εκμάθησης. Η φάση εκμάθησης διαρκεί για m βήματα. Το πλήθος m των βημάτων της φάσης εκμάθησης καθορίζεται από το χρήστη. Σε κάθε ένα από αυτά τα βήματα, ο πομπός λαμβάνει, από τους αισθητήρες, ένα νέο διάνυσμα και το στέλνει ασυμπίεστο στο δέκτη. Στο τελευταίο βήμα της φάσης εκμάθησης, υπολογίζει επιπλέον το σύνολο των κύριων συνιστωσών που αντιστοιχούν στα m διανύσματα που ελήφθησαν, χρησιμοποιώντας τη μέθοδο της ανάλυσης κύριων συνιστωσών. Από τις συνιστώσες αυτές κρατά μόνο q. Συγκεκριμένα, κρατά τις q συνιστώσες που αντιστοιχούν στις q μεγαλύτερες ιδιοτιμές. Οι συνιστώσες που θα επιλεγούν, θα χρησιμοποιηθούν, στη συνέχεια, ως η βάση για τη συμπίεση των δειγμάτων. Ο υπολογισμός του q γίνεται, όπως και στον PC3, μέσω του τύπου

$$q = \min\{q^* | \sum_{k=1}^{q^*} \frac{eigValk}{\sum_{u=1}^{n} eigValu} \geq accuracy\}, \quad (6.11)$$

όπου $eigVal_i$ είναι την i-οστή ιδιοτιμή που αντιστοιχεί στο i-οστό ιδιοδιάνυσμα του πίνακα ιδιοδιανυσμάτων, και εξαρτάται από μια παράμετρο ακρίβειας. Σε όλα τα πειράματα που κάνουμε, θέτουμε την παράμετρο ακρίβειας ίση με 0.9. Δηλαδή, θέλουμε η συμπίεση των δεδομένων με την χρήση της παραπάνω βάσης να μην οδηγεί σε απώλεια πληροφορίας μεγαλύτερη του 10%.

Μετά το τέλος της φάσης εκμάθησης, ο πομπός προβάλλει κάθε νέο δείγμα που λαμβάνει στην υπάρχουσα βάση και στη συνέχεια, το ανακατασκευάζει. Έπειτα,

υπολογίζει το σχετικό σφάλμα ανάμεσα στο αρχικό δείγμα, y, και το ανακατασκευασμένο δείγμα, y΄, το οποίο δίνεται από τον τύπο.

$$\textbf{Relative error} = |y - y'|/|y| \qquad \textbf{(6.12)}$$

Αν το σφάλμα είναι μικρότερο από κάποιο, καθορισμένο από το χρήστη, κατώφλι καινοτομίας (novelty threshold), ο πομπός στέλνει στο δέκτη το δείγμα συμπιεσμένο. Για το πρώτο βήμα μετά τη φάση της εκμάθησης, στέλνει επιπλέον και τη βάση. Αλλιώς, ο πομπός ανανεώνει, μέσω της μεθόδου αυξητικής ανάλυσης κύριων συνιστωσών, τον πίνακα των ιδιοδιανυσμάτων – βάση. Η νέα βάση που υπολογίστηκε συγκρίνεται με την παλιά. Συγκεκριμένα, υπολογίζεται η απόσταση μεταξύ των δύο βάσεων, η οποία ισούται με το άθροισμα των επιμέρους σχετικών σφαλμάτων μεταξύ των q πρώτων συνιστωσών των δύο βάσεων. Σε περίπτωση που η νέα βάση έχει επεκταθεί με μια επιπλέον συνιστώσα, αυτή δε θα ληφθεί υπόψη στον υπολογισμό της απόστασης. Αν η απόσταση των δύο βάσεων είναι μεγαλύτερη από το κατώφλι, ο πομπός στέλνει στο δέκτη το δείγμα που έλαβε ασυμπίεστο μαζί με την ανανεωμένη βάση. Αν η διαφορά των δύο βάσεων είναι μικρότερη από το κατώφλι και το πλήθος των κύριων συνιστωσών δεν έχει αλλάξει, πράγμα που σημαίνει ότι η νέα μέτρηση δεν επηρέασε πολύ τα δεδομένα και η παλιά βάση επιτρέπει τη συμπίεση τους χωρίς μεγάλη απώλεια πληροφορίας, ο πομπός στέλνει στο δέκτη το νέο δείγμα που έλαβε ασυμπίεστο. Σε αυτή την περίπτωση, ο πομπός δε χρειάζεται να στείλει στο δέκτη και την παλιά βάση, παρά μόνο για το πρώτο δείγμα που λαμβάνει μετά την φάση της εκμάθησης. Αλλιώς, αν ναι μεν η διαφορά των δύο βάσεων είναι μικρότερη από το κατώφλι, αλλά το πλήθος των κύριων συνιστωσών έχει αυξηθεί, ο πομπός στέλνει στο δέκτη το δείγμα που έλαβε ασυμπίεστο, το ιδιοδιάνυσμα που προστέθηκε στο σύνολο των κύριων συνιστωσών και αν βρισκόμαστε στο (m+1) – οστό βήμα, την παλιά βάση.

Σε κάθε βήμα του αλγορίθμου υπολογίζεται η συνολική ενέργεια η οποία έχει δαπανηθεί μέχρι το αυτό βήμα. Η ενέργεια αυτή περιλαμβάνει την ενέργεια συμπίεσης των δεδομένων στον πομπό και την ενέργεια αποσυμπίεσης των δεδομένων στο δέκτη, όταν έχουμε συμπίεση και την ενέργεια που απαιτείται για την αποστολή και τη λήψη των δεδομένων. Η μεταφορά των δεδομένων γίνεται σε πακέτα με 7 bytes επικεφαλίδα και 20 bytes πληροφορία. Στην περίπτωση που έχουμε υπολογισμό βάσης, στη συνολική ενέργεια προστίθεται και το κόστος υπολογισμού της. Ακόμα, κάθε φορά που αποστέλλεται μια βάση, στη συνολική ενέργεια προστίθεται η ενέργεια που απαιτείται για την αποστολή και τη λήψη της βάσης. Η μεταφορά της βάσης γίνεται, επίσης, με πακέτα. Πιο αναλυτικά, σε κάθε ένα από τα m βήματα της φάσης εκμάθησης προστίθεται στη συνολική ενέργεια η ενέργεια αποστολής του δείγματος από τον πομπό και η ενέργεια λήψης του δείγματος από το δέκτη. Στα βήματα για τα οποία το σχετικό σφάλμα είναι μικρότερο από το κατώφλι καινοτομίας, οπότε ο πομπός στέλνει στο δέκτη το δείγμα που έλαβε αφού το συμπιέσει, στη συνολική ενέργεια προστίθεται η ενέργεια συμπίεσης και αποσυμπίεσης του δείγματος και η ενέργεια αποστολής και λήψης του δείγματος. Αν το βήμα είναι το (m+1) – οστό, στη συνολική ενέργεια προστίθεται και η ενέργεια αποστολής και λήψης της βάσης. Στα βήματα για τα οποία το σχετικό σφάλμα ξεπερνά το κατώφλι καινοτομίας, προστίθεται στη συνολική ενέργεια το κόστος υπολογισμού της ανανεωμένης βάσης. Αν η απόσταση της παλιάς βάσης με την ανανεωμένη βάση είναι μεγαλύτερη από το κατώφλι, οπότε ο πομπός στέλνει στο δέκτη το δείγμα ασυμπίεστο και την νέα βάση, προστίθεται στη συνολική ενέργεια η ενέργεια αποστολής και λήψης του δείγματος και η ενέργεια αποστολής και λήψης της βάσης. Αν η απόσταση της παλιάς βάσης με την ανανεωμένη βάση είναι μικρότερη από το κατώφλι, αλλά το πλήθος συνιστωσών έχει αυξηθεί, προστίθεται στη συνολική ενέργεια η ενέργεια αποστολής και λήψης του δείγματος και η ενέργεια αποστολής και λήψης της νέας κύριας συνιστώσας. Αν η απόσταση της παλιάς βάσης με την ανανεωμένη βάση

είναι μικρότερη από το κατώφλι και το πλήθος των συνιστωσών δεν άλλαξε, προστίθεται στη συνολική ενέργεια η ενέργεια αποστολής και λήψης του δείγματος. Στις δυο τελευταίες περιπτώσεις, αν βρισκόμαστε στο (m+1) – οστό βήμα, στη συνολική ενέργεια προστίθεται και η ενέργεια αποστολής και λήψης της βάσης.

Σε κάθε βήμα υπολογίζεται, επίσης, το μέσο σχετικό σφάλμα που προκύπτει από την ανακατασκευή του αρχικού διανύσματος. Παρακάτω δίνεται ο αλγόριθμος σε μορφή ψευδοκώδικα.

Είσοδοι: m (ορίζοντας φάσης εκμάθησης), acur (ακρίβεια συμπίεσης), noveltyThres (κατώφλι)

curSample = 0

1. Για i από 1 μέχρι m
   α. Στείλε το διάνυσμα στο δέκτη χωρίς συμπίεση
   β. Αν i == m
      Υπολόγισε τη βάση με τη μέθοδο PCA
2. curSample = m+1
3. Μέχρι να τελειώσουν οι παρατηρήσεις
   α. Βρες την προβολή του διανύσματος επάνω στην παλιά βάση
   β. Ανακατασκεύασε το διάνυσμα
   γ. Υπολόγισε το σχετικό σφάλμα ανάμεσα στο αρχικό διάνυσμα και το ανακατασκευασμένο διάνυσμα.
   δ. Αν το σφάλμα > novelthyThres
      i. Υπολόγισε τη νέα βάση με τη μέθοδο incremental PCA
      ii. Υπολόγισε την απόσταση ανάμεσα την παλιά και τη νέα βάση
      iii. Αν η απόσταση < novelthyThres && νέος #PCs > παλιός #PCs
      - Στείλε το διάνυσμα στο δέκτη χωρίς συμπίεση.
      - Στείλε το νέο PC στο δέκτη.
      - Αν curSample == m+1
        Στείλε στο δέκτη τη βάση.

      iv. Αν η απόσταση < novelthyThres && νέος #PCs == παλιός #PCs
      - Στείλε το διάνυσμα στο δέκτη, αφού το συμπιέσεις.
      - Αν curSample == m+1
        Στείλε στο δέκτη τη βάση.

      v. Αν η απόσταση < novelthyThres
      - Στείλε το διάνυσμα στο δέκτη χωρίς συμπίεση.
      - Στείλε τη νέα βάση στο δέκτη.

   ε. Αν το σφάλμα <= novelthyThres
      i. Στείλε το διάνυσμα στο δέκτη, αφού το συμπιέσεις.
      ii. Αν curSample == m+1
          Στείλε στο δέκτη τη βάση.
   στ. curSample = curSample + 1

Δεν είναι εύκολο να κάνουμε μια σαφή πρόβλεψη για την απόδοση του incPCA. καθώς η συμπεριφορά εξαρτάται σε μεγάλο βαθμό από το είδος των δεδομένων. Στις περιπτώσεις που το δείγμα μας έχει αρχίσει να «σταθεροποιείται», είτε η διαφορά ανάμεσα στο πραγματικό διάνυσμα και το διάνυσμα που προέκυψε από την ανακατασκευή του ύστερα από προβολή στην παλιά βάση θα είναι μικρή και επομένως, θα κάνουμε συμπίεση, είτε η απόσταση της παλιάς με τη νέα βάση θα είναι μικρή και θα χρειάζεται να γίνει αποστολή μόνο του συμπιεσμένου δείγματος ή να γίνει αποστολή του δείγματος και της νέας κύριας συνιστώσας. Τα παραπάνω θα έχουν ως αποτέλεσμα τη μείωση της συνολικής ενέργειας που πρέπει να δαπανηθεί. Στις περιπτώσεις, τώρα, όπου τα δείγματα αλλάζουν κατά πολύ τη βάση, ναι μεν θα δαπανάται περισσότερη ενέργεια, αλλά το μέσο σφάλμα θα μικραίνει. Σε σύγκριση με τους άλλους 3 αλγορίθμους, θεωρούμε ότι ο incPCA θα έχει καλύτερο μέσο σφάλμα, λόγω των συχνών ανανεώσεων που κάνει στη βάση.

7. ΠΕΙΡΑΜΑΤΑ, ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο κεφάλαιο αυτό, έχοντας πλέον περιγράψει πλήρως τα νέα προτεινόμενα σχήματα, θα συγκρίνουμε τα σχήματα αυτά μεταξύ τους και με τον πρωταρχικό μας PC3 [1]. Η σύγκριση θα γίνει πάνω σε ένα κοινό σύνολο δεδομένων και πέρα από την κατανάλωση ενέργειας και το μέσο συνολικό σχετικό σφάλμα, βάσει κάποιων άλλων μετρικών που θα ορίσουμε στη συνέχεια.

7.1 Το πειραματικό σύνολο δεδομένων και το μοντέλο κατανάλωσης ενέργειας

Τα πειράματα που θα δούμε παρακάτω βασίστηκαν σε ένα πειραματικό σύνολο από 3870 διανυσματικές μετρήσεις, που είναι αποτέλεσμα της αναπαραγωγής και μικρής αλλαγής 387 πραγματικών μετρήσεων από αισθητήρες. Η κάθε μία διανυσματική μέτρηση αποτελείται από 7 μεταβλητές και συμβολίζεται με:

x *= (temp1 , hum1 , temp2 , hum2 , temp3 , hum3 , wind4)* .

Οι πρώτες έξι μετρήσεις ανά δυάδες είναι ζεύγη μετρήσεων θερμοκρασίας και υγρασίας, ενώ η έβδομη μέτρηση είναι μια μέτρηση της ταχύτητας του ανέμου.

Σε ό,τι αφορά την κατανάλωση ενέργειας, υιοθετήσαμε το μοντέλο κατανάλωσης ενέργειας Mica2 [6], όπως έπραξαν και οι Anagnostopoulos et al. στο [1].Τα ενεργειακά κόστη για μοναδικές εντολές και για τη μετάδοση και τη λήψη τιμών πλαισίου παρουσιάζονται στον πίνακα 7.1 που ακολουθεί.

Πίνακας 7.1 Ενεργειακά κόστη

Λειτουργία κόμβου	**Ενεργειακό κόστος**
Εκτέλεση εντολής	4nJ/εντολή
Ανενεργός	9,6 mJ/s
Σε αναμονή	0,33 mJ/s
Εκπομπή	720nJ/bit
Λήψη	110nJ/bit

7.2 Παράμετροι των προτεινόμενων σχημάτων

Όλα τα προτεινόμενα σχήματα απαιτούν m χρονικές μονάδες για την εκμάθηση του n x q πίνακα ιδιοδιανυσμάτων. Ο αριθμός των κυρίων συνιστωσών, $1 \leq q \leq n$, ο οποίος εξαρτάται από το ποσοστό ακρίβειας, θα ποικίλει μεταξύ δυο περιόδων συμπίεσης. Σε όλα τα πειράματα, όπως έχουμε ήδη αναφέρει, θέτουμε την ακρίβεια ίση με 0.9. Δηλαδή, ζητάμε πιστότητα στη συμπίεση της τάξης του 90%. Για τον PC3 χρειάζεται επίσης να δοθεί τιμή στην παράμετρο l που αναπαριστά τον ορίζοντα διάδοσης της φάσης συμπίεσης. Για τον infHorPC31 χρειάζεται να δοθεί τιμή στην παράμετρο errorThres, η οποία αναπαριστά το κατώφλι σφάλματος με το οποίο συγκρίνεται σε κάθε βήμα της φάσης συμπίεσης το ποσοστιαίο σφάλμα ανακατασκευής του τρέχοντος δείγματος στον δέκτη και στη c, η οποία αναπαριστά το κόστος ανά δείγμα. Για τον infHorPC32 χρειάζεται να δοθεί τιμή στην παράμετρο errorThres, η οποία αναπαριστά και εδώ το κατώφλι σφάλματος με το οποίο συγκρίνεται σε κάθε βήμα της φάσης συμπίεσης το ποσοστιαίο σφάλμα ανακατασκευής του τρέχοντος δείγματος στον δέκτη

και στη b, η οποία αναπαριστά την πιθανότητα να μη χάσουμε το κέρδος που έχουμε συσσωρεύσει σε κάθε βήμα. Τέλος, ο incPCA παίρνει επίσης μια επιπρόσθετη παράμετρο τη noveltyThres. Βάσει της παραμέτρου αυτής, όπως είδαμε, αποφασίζουμε αν θα στείλουμε κάποιο διάνυσμα μετά τη φάση εκμάθησης συμπιεσμένο ή ασυμπίεστο και αν το στείλουμε ασυμπίεστο, αν προκαλεί τέτοια αλλαγή στην παλιά βάση που μας αναγκάζει να την αναθεωρήσουμε κρατώντας την νέα βάση που υπολογίζουμε. Για την τελική επιλογή των τιμών των παραμέτρων πραγματοποιήσαμε κάποια αρχικά πειράματα με διαφορετικούς συνδυασμούς τιμών στις παραμέτρους για τους ίδιους αλγορίθμους. Από κάθε αλγόριθμο θέλαμε να επιλέξουμε τις τιμές των παραμέτρων που θα έδιναν μεγαλύτερο λόγο (συνολική ενέργεια συμπίεσης)/(μέσο συνολικό σχετικό σφάλμα). Παρατηρήσαμε ότι ο PC3 είχε καλύτερη απόδοση για m = 5 και l=10, επειδή, όμως, όλοι οι άλλοι είχαν καλύτερη απόδοση για m = 7, αποφασίσαμε, τελικά, να τους συγκρίνουμε όλους με κοινό m, επιλέγοντας το 7. Εκεί, πάλι, ο PC3 είχε καλύτερη απόδοση για l = 10. Ακολουθούν οι τελικές τιμές που επιλέχθηκαν για κάθε αλγόριθμο.

- PC3 (m = 7, l = 10)
- incPCA (m = 7, noveltyThres = 0.02)
- infHorPC31 (m = 7, errorThres = 0.01, c = 0.00001)
- infHorPC32 (m = 7, errorThres = 0.01, b = 0.8)

Εκτός από αυτές τις τιμές, όμως, συγκρίναμε τους αλγορίθμους PC3 και infHorPC32 για m = 5, l = 10, errorThres = 0.01 και b = 0.8. Επίσης, πραγματοποιήσαμε πειράματα που δείχνουν τη μεταβολή διαφόρων μεγεθών για διαφορετικές τιμές του ορίζοντα της φάσης εκμάθησης m, διατηρώντας σταθερές τις τιμές των υπολοίπων παραμέτρων. Συνολικά, οι τιμές των παραμέτρων που χρησιμοποιούμε στα πειράματά μας φαίνονται στον πίνακα 7.2 που ακολουθεί.

Πίνακας 7.2 Τιμές παραμέτρων για την προσομοίωση

Διάσταση διανύσματος πλαισίου	7
Διάρκεια φάσης εκμάθησης m	[2, 10]
Διάρκεια φάσης συμπίεσης l	10
Ποσοστό ακρίβειας a	90%
Κατώφλι σφάλματος errorThres	[0.01, 0.05]
Κόστος ανά δείγμα c	0.00001
Πιθανότητα διατήρησης κέρδους b	[0.5, 0.9]
Κατώφλι καινοτομίας noveltyThres	[0.02, 0.1]

7.3 Χρησιμοποιούμενες μετρικές επίδοσης

Όλα τα προτεινόμενα σχήματα μαζί και τον PC3 τα συγκρίναμε ως προς τα παρακάτω μέτρα:

- Energy Consumption, που είναι η συνολική κατανάλωση ενέργειας μέχρι κάποιο χρονικό σημείο, π.χ., στο (m-1) βήμα της learning phase η κατανάλωση ενέργειας θα είναι ίση με (m-1)*(κόστος αποστολής + κόστος λήψης) ενός ασυμπίεστου διανύσματος διάστασης n, για όλους τους αλγορίθμους. Σημειώνουμε ότι στην κατανάλωση ενέργειας λαμβάνουμε υπόψη και την ενέργεια που δαπανάται για τον υπολογισμό της βάσης μέσω της μεθόδου PCA και επιπλέον, στον incPCA και την ενέργεια για τον υπολογισμό των ιδιοδιανυσμάτων που θα αποτελέσουν τη νέα βάση.

- Energy Gain, που είναι το ενεργειακό κέρδος μέχρι κάποια χρονική μονάδα. Το ενεργειακό κέρδος ορίζεται ως εξής:

$$\frac{(\textbf{συνολική ενέργεια αποστολής/λήψης ασυμπίεστων δειγμάτων} - \textbf{EnergyConsumption})}{(\textbf{συνολική ενέργεια αποστολής/λήψης ασυμπίεστων δειγμάτων})} \quad \textbf{(7.1)}$$

Το τελικό ενεργειακό κέρδος είναι το ενεργειακό κέρδος στο τελευταίο δείγμα.

- Efficiency, που είναι η απόδοση. Για την απόδοση προτείνουμε δύο μετρικές. Η πρώτη ορίζεται ως ο λόγος του energy gain σε κάθε βήμα δια το μέσο συνολικό σχετικό σφάλμα (mean total relative error ή mtre) μέχρι το βήμα αυτό. Δηλαδή,

$$\textbf{Efficiency} = \frac{\textit{Energy Gain}}{\textit{mean total relative error}} . \quad \textbf{(7.2)}$$

Η τελική απόδοση είναι η απόδοση στο τελευταίο βήμα. Μπορεί κανείς να παρατηρήσει ότι ο παραπάνω λόγος δεν ορίζεται για όλες τις χρονικές μονάδες, εφόσον π.χ., για τον PC3 στις πρώτες (m -1) χρονικές μονάδες τα διανύσματα στέλνονται ασυμπίεστα, επομένως τόσο το ενεργειακό κέρδος όσο και το συνολικό σχετικό σφάλμα θα είναι μηδενικά. Ακόμη, τη χρονική στιγμή m όπου γίνεται ο υπολογισμός της βάσης και στέλνεται ασυμπίεστο το m – οστό διάνυσμα, αν και το ενεργειακό κέρδος έχει μη μηδενική και ελαφρώς αρνητική τιμή, το σφάλμα εξακολουθεί να είναι μηδενικό. Για το λόγο αυτό κατά τα (m – 1) πρώτα βήματα όλων των σχημάτων θεωρούμε την απόδοση μηδενική, ενώ για το m – οστό βήμα τη θεωρούμε ίση με το ενεργειακό κέρδος. Η δεύτερη μετρική ορίζεται ως εξής:

$$\textbf{New Efficiency Metric = 0.5 *Energy Gain + 0.5 * (1 – mean total Relative Error)}. \quad \textbf{(7.3)}$$

Είναι προφανές ότι η δεύτερη μετρική μπορεί να πάρει ως μέγιστη τιμή την τιμή 1 και αυτό θα γινόταν στην περίπτωση όπου θα είχαμε ενεργειακό κέρδος ίσο με 100% και πιστότητα στα δεδομένα ίση με 100%. Με τη δεύτερη μετρική το ενεργειακό κέρδος και η πιστότητα στα δεδομένα έχουν το ίδιο βάρος στη μέτρηση της επίδοσης ενός σχήματος.

- Compression ratio, που είναι το ποσοστό συμπίεσης και ορίζεται ως ο λόγος των δειγμάτων που στέλνονται συνολικά κατά τη συμπίεση δια τον αριθμό των δειγμάτων που στέλνονται χωρίς καμία συμπίεση, μέχρι μια ορισμένη χρονική μονάδα. Δηλαδή,

$$\textbf{Compression ratio} = \frac{\textit{\# συμπιεσμένων δειγμάτων}}{\textit{\# δειγμάτων χωρίς συμπίεση}}. \quad \textbf{(7.4)}$$

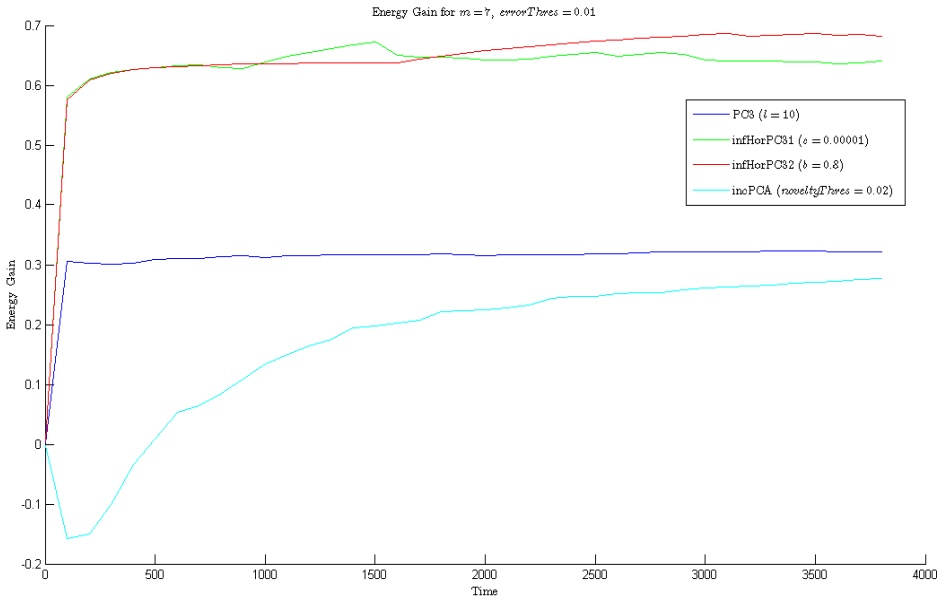
Όσο μικρότερη τιμή θα έχει το compression ratio, τόσο πιο αποδοτικός θα είναι ο αλγόριθμος στη μετάδοση μικρού αριθμού διανυσμάτων πλαισίου διατηρώντας παράλληλα το παραγόμενο σφάλμα σε χαμηλά επίπεδα.

Επιπλέον, για τη σύγκριση των αλγορίθμων με μη πεπερασμένο ορίζοντα διάδοσης μεταξύ τους μετράμε για διαφορετικό μήκος της φάσης εκμάθησης πόσες είναι κάθε φορά οι περίοδοι συμπίεσης και επιπρόσθετα πόσο είναι το μέσο μήκος των οριζόντων συμπίεσης. Σαφώς, όσο μεγαλύτερες σε μήκος θα είναι οι περίοδοι συμπίεσης, τόσο λιγότερες θα είναι σε αριθμό. Αντίστροφα, όσο αυξάνεται το πλήθος των περιόδων συμπίεσης, τόσο θα μειώνεται ο μέσος ορίζοντας συμπίεσης.

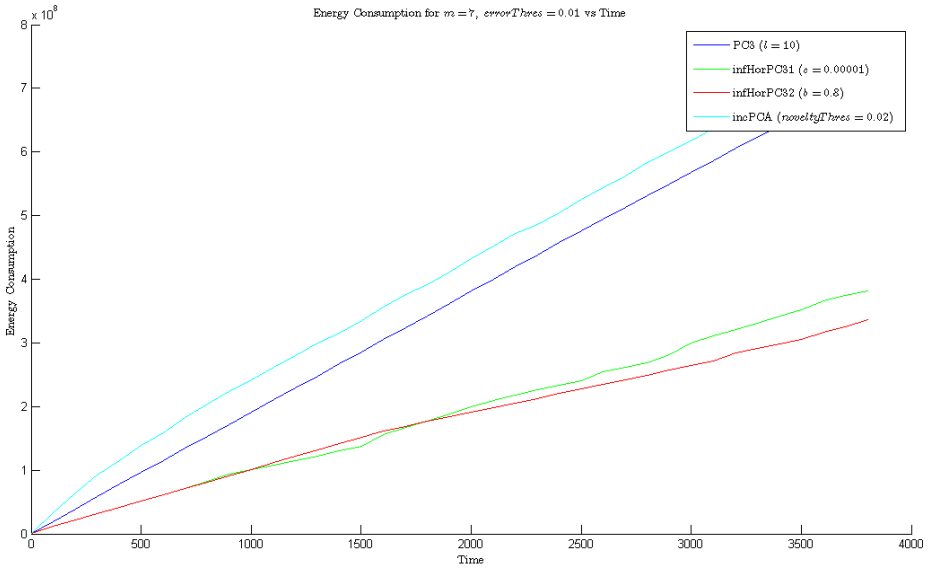
7.4 Αποτίμηση επίδοσης

Στην παράγραφο αυτή μελετούμε την επίδοση των προτεινόμενων σχημάτων σε σχέση με τις προαναφερθείσες μετρικές απόδοσης, δίνοντας και διαγράμματα που δείχνουν τη συγκριτική επίδοση των αλγορίθμων μεταξύ τους. Αρχικά, θα δούμε τη συμπεριφορά των σχημάτων ως προς την κατανάλωση ενέργειας και το ενεργειακό κέρδος, στη συνέχεια τη συμπεριφορά τους ως προς το επίπεδο σφάλματος και το ποσοστό συμπίεσης και τέλος, θα δούμε τη συνολική απόδοση των αλγορίθμων βάσει της μετρικής της απόδοσης που ορίσαμε στην παράγραφο 7.3. Επιπλέον, για την επιμέρους σύγκριση των αλγορίθμων infHorPC31 και infHorPC32, θα δούμε διαγράμματα που δείχνουν το πλήθος των φάσεων συμπίεσης και το μέσο ορίζοντα συμπίεσης για διαφορετικές τιμές της παραμέτρου m.

7.4.1 Κατανάλωση ενέργειας και ενεργειακό κέρδος

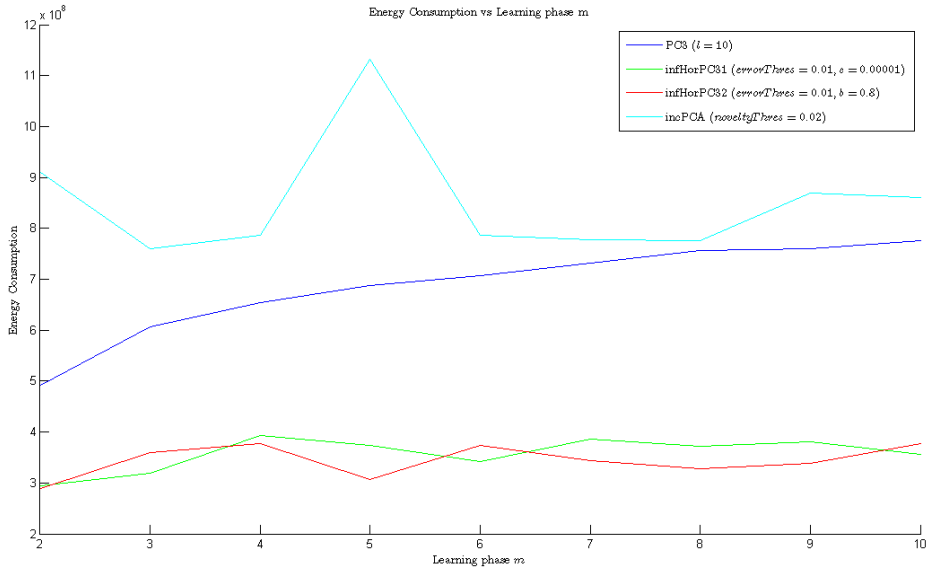


Διάγραμμα 7.1 Μεταβολή ενεργειακού κέρδους ως προς το χρόνο για μήκος φάσης εκμάθησης 7 και κατώφλι σφάλματος errorThres=1% για όλους τους αλγορίθμους.

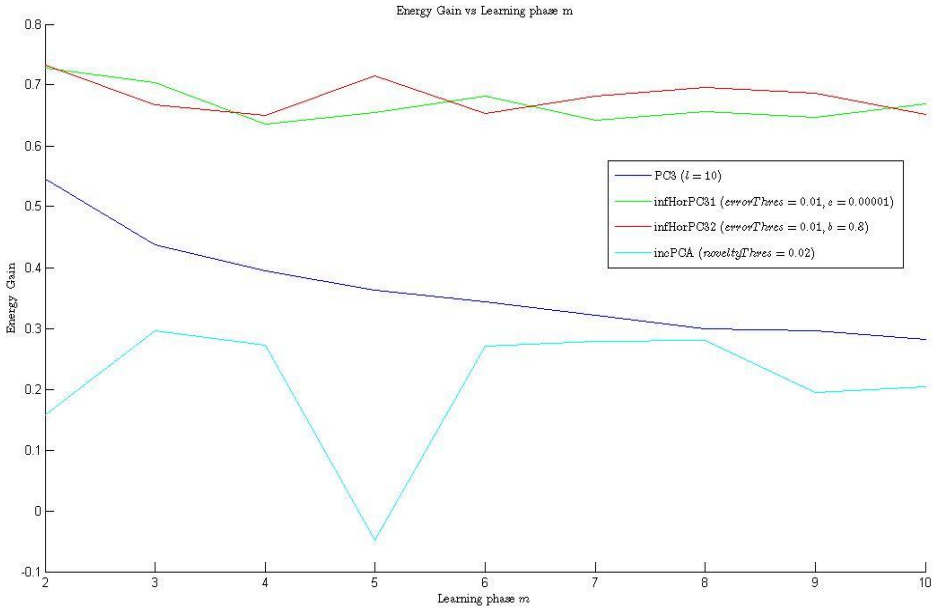


Διάγραμμα 7.2 Μεταβολή της κατανάλωσης ενέργειας ως προς το χρόνο για μήκος φάσης εκμάθησης m=7 και κατώφλι σφάλματος 1% για όλους τους αλγορίθμους.

Όσον αφορά την κατανάλωση ενέργειας και αντίστοιχα το ενεργειακό κέρδος για m = 7, βλέπουμε ότι ο incPCA (noveltyThres = 0.02) έχει τη μεγαλύτερη κατανάλωση ενέργειας και άρα, το μικρότερο ενεργειακό κέρδος. Ακολουθεί ο PC3 (l = 10), μετά ο infHorPC31 (errorThres = 0.01, c = 0.00001) και τελευταίος είναι ο infHorPC32 (errorThres = 0.01, b = 0.8). Άρα, μεγαλύτερο ενεργειακό κέρδος έχει ο infHorPC32. Το αποτέλεσμα αυτό ήταν αναμενόμενο, εφόσον ο infHorPC32 τερματίζει τη φάση συμπίεσης με χρήση ενός κανόνα βέλτιστης παύσης, ο οποίος μάλιστα για κάθε νεοεισερχόμενο δείγμα μεταβάλλεται ώστε να λάβει και το νέο αυτό δείγμα υπόψη στην απόφαση για την παύση. Επίσης, λογικό είναι ότι από άποψη ενεργειακού κέρδους ακολουθεί ο infHorPC31, εφόσον και αυτός χρησιμοποιεί έναν κανόνα βέλτιστης παύσης για τη λήξη της φάσης της συμπίεσης. Το μικρό προβάδισμα του infHorPC32 έναντι του infHorPC31 οφείλεται στο γεγονός ότι ο δεύτερος για τη λήψη της απόφασης παύσης με τον ερχομό ενός νέου διανύσματος συγκρίνει το θεωρούμενο τρέχον σφάλμα με έναν παράγοντα που δε μεταβάλλεται, αλλά παραμένει σταθερός καθ' όλη τη φάση της συμπίεσης και είναι ίσος, όπως είδαμε, με $\sqrt{(2 * c)}$. Σε ό,τι αφορά τον incPCA, επίσης το αποτέλεσμα είναι το προσδωκόμενο, εφόσον ο incPCA υπολογίζει για κάθε νεοεισερχόμενο διάνυσμα με μέσο ποσοστιαίο σφάλμα ανακατασκευής μεγαλύτερο του noveltyThres τα νέα ιδιοδιανύσματα που πιθανόν να αποτελέσουν τη νέα μας βάση, πράγμα που έχει εκθετικό κόστος ως προς τον αριθμό των ιδιοδιανυσμάτων. Έτσι, συχνά ο incPCA μπορεί να δαπανά περισσότερη ενέργεια από αυτή που θα καταναλωνόταν αν στέλναμε όλα τα διανύσματα ασυμπίεστα. Αυτό φαίνεται σαφέστατα στο διάγραμμα, όπου μέχρι και σχεδόν τα 500 πρώτα δείγματα το ενεργειακό του κέρδος είναι αρνητικό. Βέβαια, επειδή ανανεώνει συχνά τη βάση, ο αλγόριθμος αυτός θα οδηγήσει σε πολύ μικρότερο σφάλμα συγκριτικά με τους άλλους αλγορίθμους, όπως θα δούμε παρακάτω.



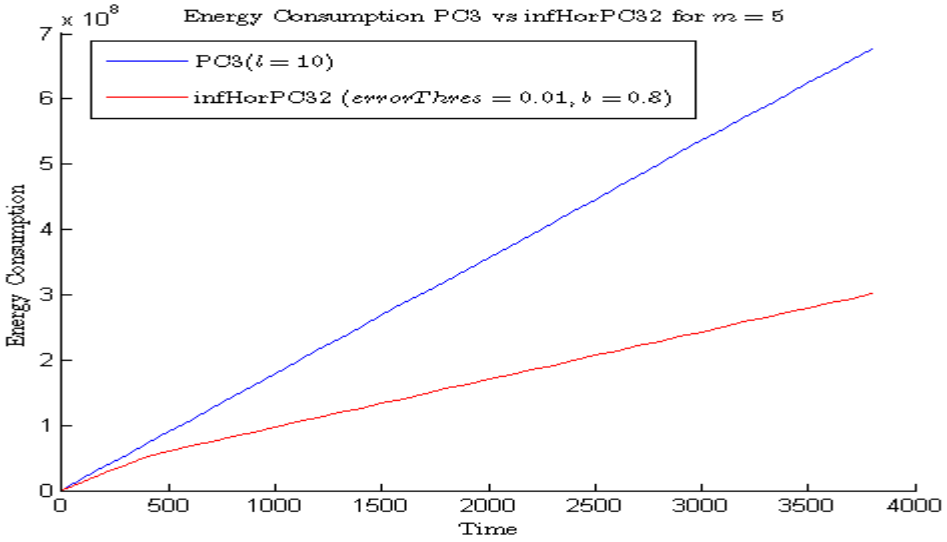
Διάγραμμα 7.3 Μεταβολή της κατανάλωσης ενέργειας ως προς το μήκος της φάσης εκμάθησης για όλους τους αλγορίθμους



Διάγραμμα 7.4 Μεταβολή του ενεργειακού κέρδους ως προς το μήκος της φάσης εκμάθησης για όλους τους αλγορίθμους.

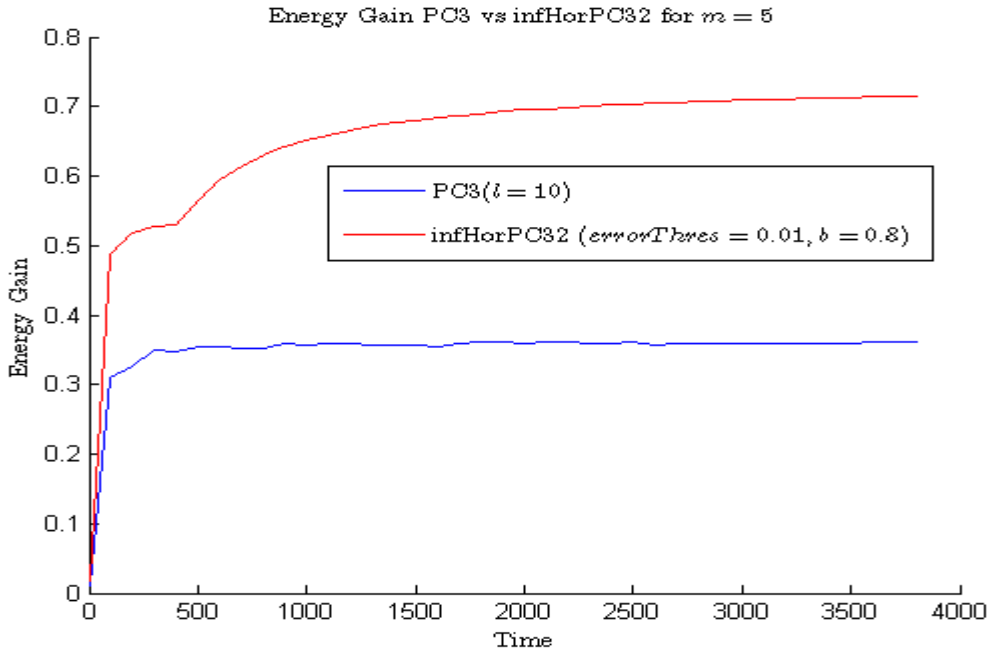
Στο παραπάνω διάγραμμα παρατηρούμε πως μεταβάλλεται το συνολικό ενεργειακό κέρδος για διαφορετικές τιμές της παραμέτρου m της φάσης εκμάθησης. Το μόνο ασφαλές συμπέρασμα που μπορούμε να εξάγουμε από εδώ είναι ότι ο PC3 καταναλώνει περισσότερη ενέργεια όσο αυξάνεται η τιμή της παραμέτρου m. Αυτό είναι λογικό, εφόσον για μεγαλύτερο m θα στέλνει περισσότερα διανύσματα ασυμπίεστα. Το ενεργειακό κέρδος των infHorPC31 και infHorPC32 είναι υψηλότερο από αυτό των PC3 και incPCA για όλα τα m και του incPCA είναι το χαμηλότερο από όλα. Επίσης, οι infHorPC31 και infHorPC32 για καμία άλλη τιμή του m δεν πετυχαίνουν τόσο μεγάλο ενεργειακό κέρδος όσο για m = 2. Ενδιαφέρον παρουσιάζει το γεγονός ότι για m = 5, το

ενεργειακό κέρδος του incPCA πέφτει απότομα. Αυτό πιθανότατα οφείλεται στη φύση των πειραματικών δεδομένων.



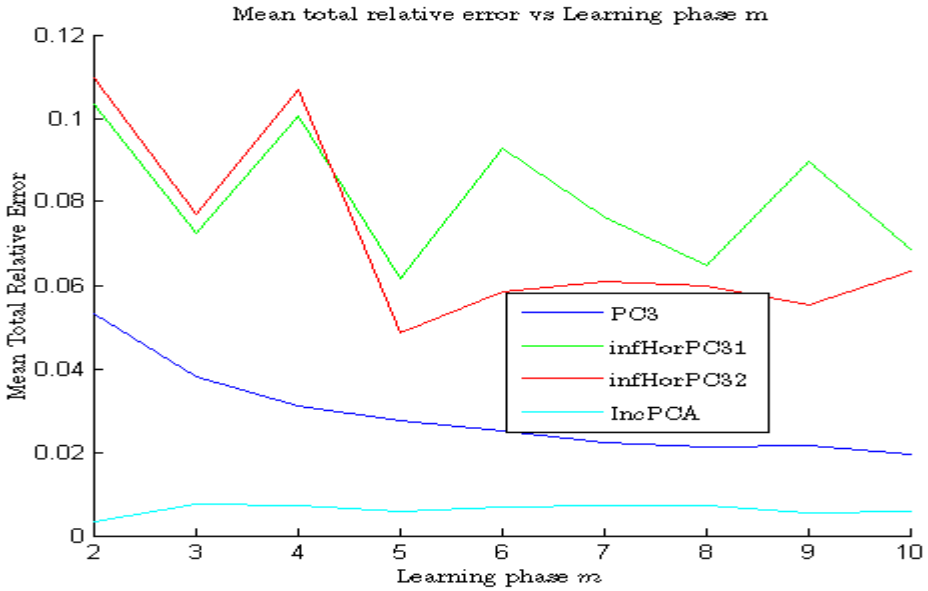
Διάγραμμα 7.5 Μεταβολή της κατανάλωσης ενέργειας ως προς το χρόνο για τους αλγορίθμους PC3 και infHorPC32 για μήκος φάσης εκμάθησης m=5.

Βλέπουμε ότι ο infHorPC32 για m = 5 έχει σε όλες τις χρονικές μονάδες λιγότερη κατανάλωση ενέργειας και επομένως, μεγαλύτερο ενεργειακό κέρδος, όπως φαίνεται στο παρακάτω διάγραμμα. Τα διαγράμματα αυτά θα μας χρησιμεύσουν, σε συνδυασμό με τα διαγράμματα της απόδοσης για τους ίδιους δύο αλγορίθμους και τις ίδιες τιμές στις παραμέτρους, για την εξαγωγή συμπεράσματος, όπου αυτό είναι εφικτό, σχετικά με το μέσο συνολικό σχετικό σφάλμα των αλγορίθμων.



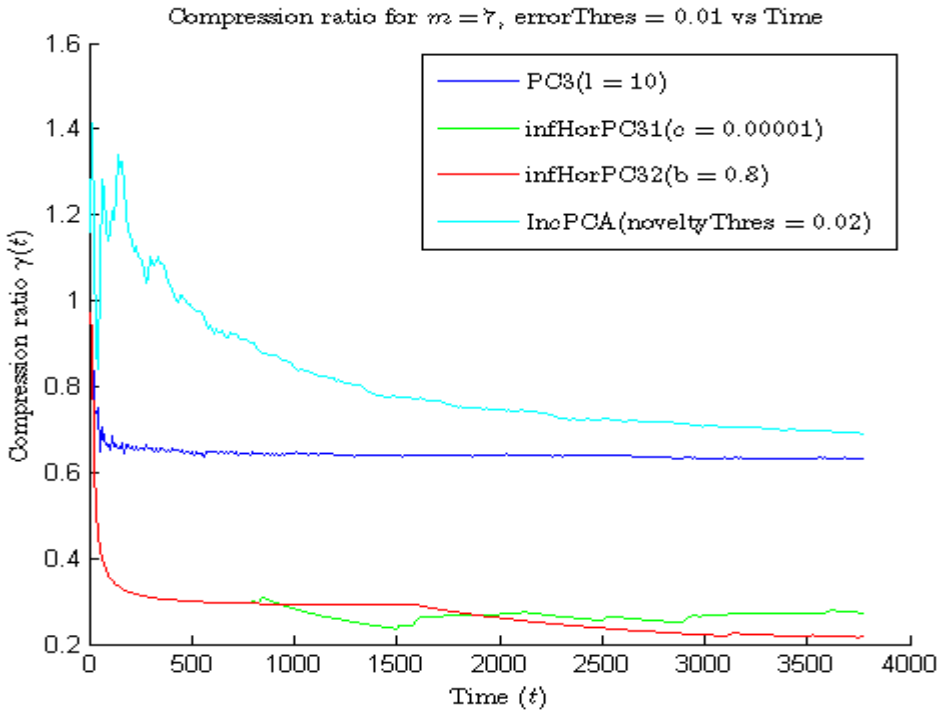
Διάγραμμα 7.6 Μεταβολή του ενεργειακού κέρδους ως προς το χρόνο για τους αλγορίθμους PC3 και infHorPC32 για μήκος φάσης εκμάθησης 5.

7.4.2 Μέσο συνολικό σχετικό σφάλμα και ποσοστό συμπίεσης



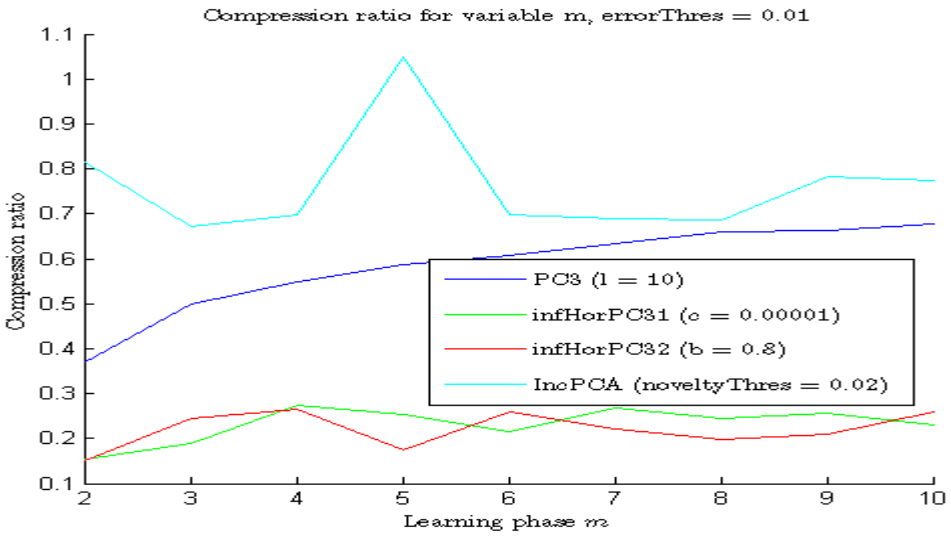
Διάγραμμα 7.7 Μεταβολή συνολικού σχετικού σφάλματος ως προς το μήκος της φάσης εκμάθησης για όλους τους αλγορίθμους.

Όπως βλέπουμε στο διάγραμμα 7.7, η συμπεριφορά των αλγορίθμων ως προς το μέσο συνολικό σχετικό σφάλμα για διαφορετικό m είναι αντιστρόφως ανάλογη της συμπεριφοράς τους ως προς την κατανάλωση ενέργειας. Έτσι, εδώ, έχουμε τον incPCA να πετυχαίνει το μικρότερο σφάλμα, συγκριτικά με όλους τους υπόλοιπους, για όλα τα m. Το σφάλμα, μάλιστα, του incPCA μπορούμε να πούμε ότι μένει εν γένει σταθερό για όλες τις τιμές της παραμέτρου m και πάντα αρκετά χαμηλότερο του 1%. Η συμπεριφορά αυτή του incPCA είναι απόλυτα λογική από τη στιγμή που ανανεώνει πολύ συχνά τη βάση των ιδιοδιανυσμάτων βάσει της οποίας πραγματοποιείται η συμπίεση. Δεύτερος στη σειρά είναι ο PC3, ο οποίος πετυχαίνει αρχικά σφάλμα της τάξης του 5% και με την αύξηση της τιμής στην παράμετρο m, κατορθώνει να μειώσει σημαντικά το σφάλμα προσεγγίζοντας για m = 10 το 2%. Επίσης, το αποτέλεσμα αυτό ήταν αναμενόμενο, αφού για μεγαλύτερες τιμές του m ο πομπός στον PC3 στέλνει περισσότερα διανύσματα ασυμπίεστα στον δέκτη. Άρα, για περισσότερα δείγματα το ποσοστιαίο σφάλμα ανακατασκευής τους στον δέκτη θα είναι μηδενικό. Όσον αφορά τους infHorPC31 και infHorPC32, όσο το m κυμαίνεται μεταξύ 2 και 4, έχουν και οι δύο σφάλμα μεγαλύτερο του 10%, με τον infHorPC31 να πετυχαίνει το μικρότερο σφάλμα. Για m = 5 και οι δύο αλγόριθμοι ρίχνουν σημαντικά το σφάλμα προσεγγίζοντας ο μεν infHorPC31 το 6,5%, ο δε infHorPC32 το 5%. Το γεγονός αυτό οφείλεται στην μορφή του πειραματικού συνόλου που διαθέτουμε. Για τιμές του m μεγαλύτερες του 5 ο infHorPC32 αυξάνει το σφάλμα του, παραμένοντας, ωστόσο, κάτω από 7%, ενώ ο infHorPC31 άλλοτε αυξάνει το σφάλμα του και άλλοτε το μειώνει, παραμένοντας κάτω του 10% σε κάθε περίπτωση.



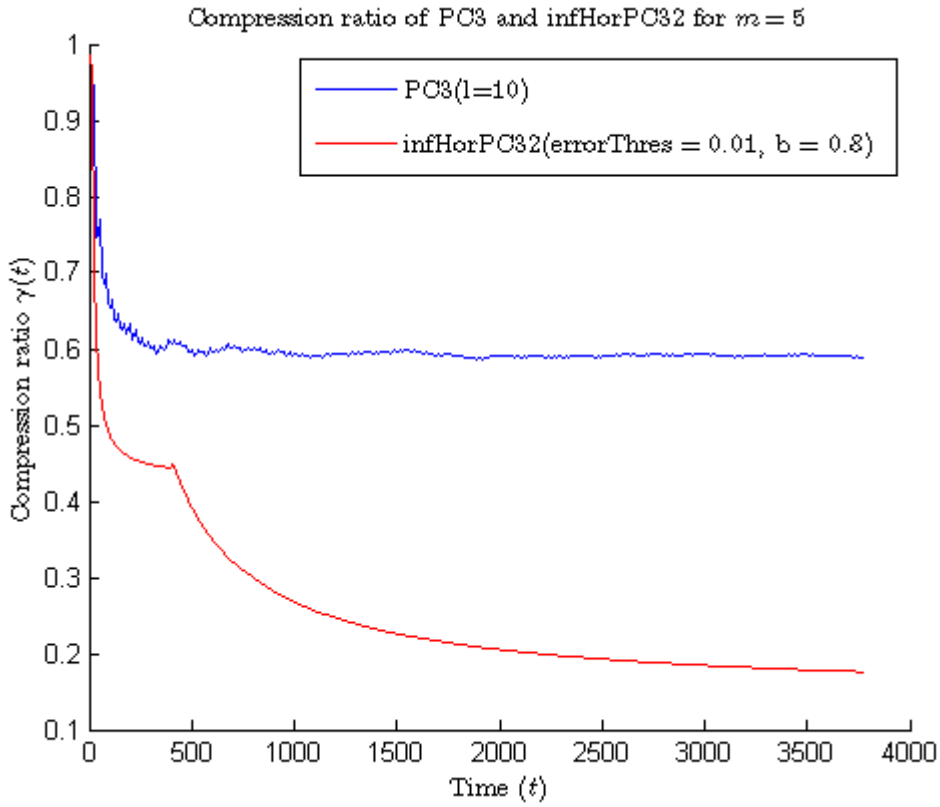
Διάγραμμα 7.8 Μεταβολή του ποσοστού συμπίεσης ως προς το χρόνο για μήκος φάσης εκμάθησης m=7 και κατώφλι σφάλματος errorThres=1% για όλους τους αλγορίθμους.

Όσον αφορά το ποσοστό συμπίεσης, το PCA έχει μεγαλύτερο compression ratio, πράγμα που σημαίνει ότι στέλνει τα περισσότερα δεδομένα. Αυτό είναι απόλυτα λογικό, αν αναλογιστούμε ότι έχει μικρότερο σφάλμα και μεγαλύτερη κατανάλωση ενέργειας. Βέβαια, το compression ratio του μειώνεται με την πάροδο του χρόνου, μένοντας σχεδόν σταθερό μετά τις 2000 χρονικές μονάδες. Αμέσως μεγαλύτερο compression ratio διαθέτει ο PC3 ο οποίος στέλνει σχεδόν το ίδιο ποσό δεδομένων μετά τις πρώτες 100 περίπου χρονικές μονάδες. Ακολουθούν οι infHorPC31 και infHorPC32 οι οποίοι στέλνουν το ίδιο ποσό δεδομένων κατά τις πρώτες 800 περίπου χρονικές μονάδες. Το ποσό των δεδομένων που στέλνει ο infHorPC32 μειώνεται με την πάροδο του χρόνου, πράγμα που εξηγεί το γεγονός ότι είναι ο αλγόριθμος με το μεγαλύτερο ενεργειακό κέρδος, όπως είδαμε προηγούμενα.



Διάγραμμα 7.9 Μεταβολή ποσοστού συμπίεσης ως προς μήκος φάσης εκμάθησης για όλους τους αλγορίθμους.

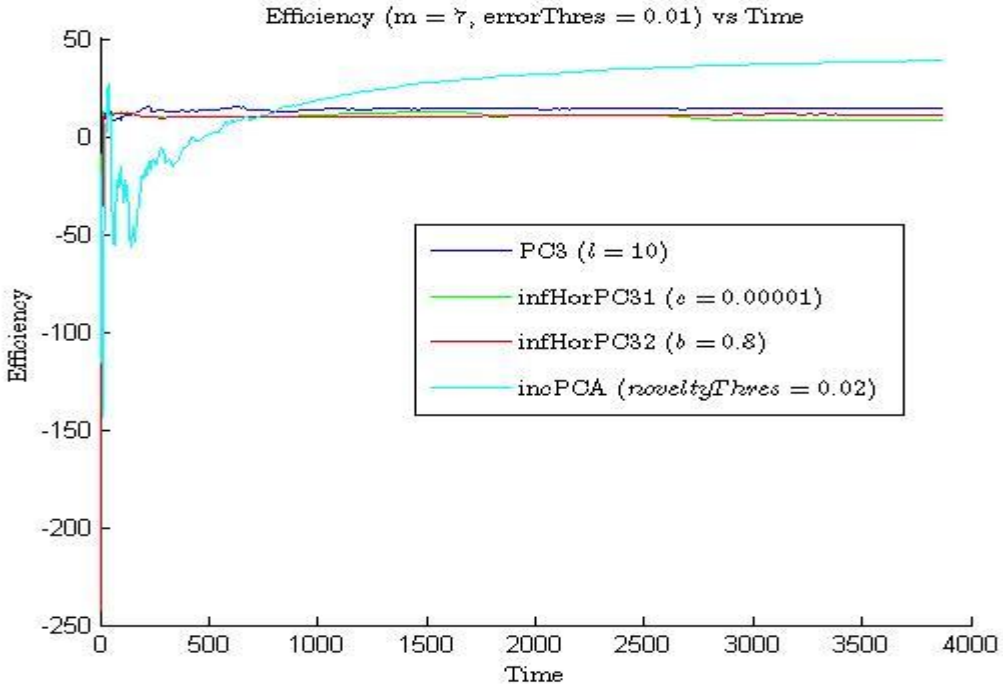
Από το διάγραμμα 7.9, παρατηρούμε ότι για όλες τις τιμές του m, ο incPCA στέλνει τα περισσότερα δεδομένα, ακολουθεί ο PC3 και μετά οι infHorPC31 και infHorPC32 οι οποίοι πετυχαίνουν τιμές συμπίεσης κοντινές μεταξύ τους. Υπάρχουν διαστήματα τιμών του m που υπερτερεί ο infHorPC32, αφού έχοντας μικρότερη τιμή compression ratio στέλνει λιγότερα δεδομένα, όπως στο διάστημα 7-9, ενώ σε άλλα διαστήματα υπερτερεί ο infHorPC31. Γενικά, για τα νέα προτεινόμενα σχήματα δεν μπορούμε να εξάγουμε ένα γενικό κανόνα για το τι συμβαίνει με την αύξηση του ορίζοντα της φάσης εκμάθησης. Το μόνο που θα μπορούσαμε να πούμε είναι ότι διαφορετικές τιμές στην παράμετρο m, οδηγούν σε διαφορετικές αρχικές βάσεις ιδιοδιανυσμάτων και διαφορετικά ποσοστιαία σφάλματα στην ανακατασκευή, τα οποία σε συνδυασμό με τις τιμές των υπολοίπων παραμέτρων και τη φύση των πειραματικών δεδομένων οδηγούν σε ανανεώσεις της βάσης σε διαφορετικές χρονικές μονάδες. Αυτό συμβαίνει επειδή για τους infHorPC31 και infHorPC32 θα μεταβληθεί ο χρόνος στον οποίο υπολογίζεται η βέλτιστη παύση, ενώ για τον incPCA θα μεταβληθεί ο χρόνος κατά τον οποίο το ποσοστιαίο σφάλμα ανακατασκευής θα ξεπεράσει το noveltyThres και η νέα βάση θα διαφέρει από την παλιά περισσότερο από noveltyThres, οπότε και θα ανανεωθεί η παλιά βάση με αποστολή ολόκληρης ή μέρους της νέας.



Διάγραμμα 7.10 Μεταβολή ποσοστού συμπίεσης ως προς το χρόνο για μήκος φάσης εκμάθησης m=5 για τους αλγορίθμους PC3 και infHorPC32.

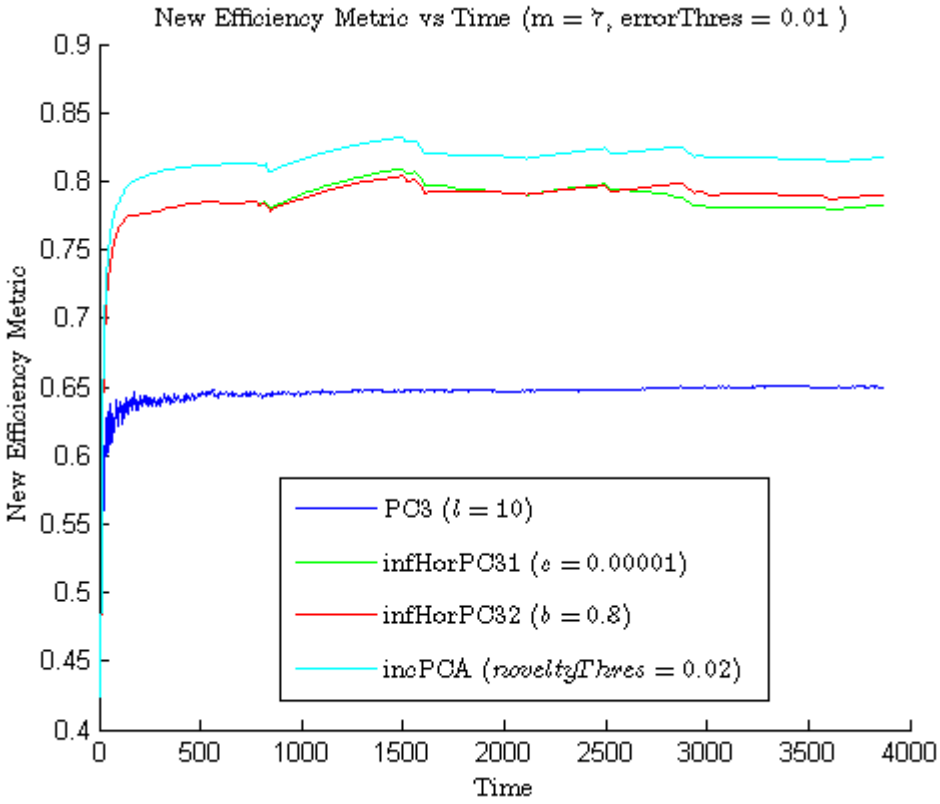
Τέλος, παρουσιάζουμε και το διάγραμμα 7.10 για να δείξουμε ότι ο λόγος για τον οποίο ο infHorPC32 υπερτερεί σημαντικά του PC3 σε ενεργειακό κέρδος για m = 5 είναι ότι έχει πολύ χαμηλότερο compression ratio, συνεπώς στέλνει πολύ λιγότερο ποσό δεδομένων σε σχέση με τον PC3.

7.4.3 Απόδοση



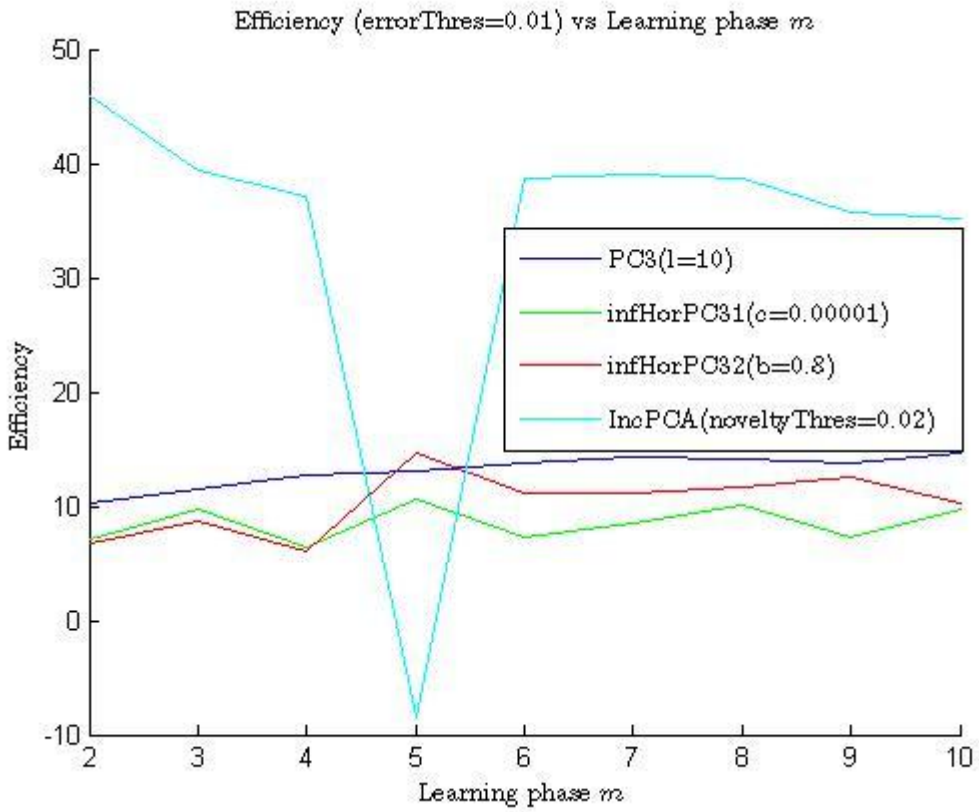
Διάγραμμα 7.11 Μεταβολή της απόδοσης ως προς το χρόνο για μήκος φάσης εκμάθησης m=7 και κατώφλι σφάλματος errorThres=1% για όλους τους αλγορίθμους.

Όσον αφορά την απόδοση για m = 7, ο incPCA από την χρονική στιγμή t≈750 και μετά είναι πάντα αποδοτικότερος από τους τρεις άλλους αλγορίθμους, όπως φαίνεται στο διάγραμμα 7.11. Η αρχική χαμηλή απόδοση του incPCA οφείλεται στο γεγονός ότι έχει μεγαλύτερη κατανάλωση ενέργειας λόγω της συχνής ανανέωσης της βάσης, άρα θα έχει και μικρότερο ενεργειακό κέρδος συγκριτικά με τους άλλους. Εφόσον, από το 750 – οστό δείγμα περίπου και μετά ο incPCA έχει καλύτερη απόδοση από τους PC3, infHorPC31 και infHorPC32, ενώ το ενεργειακό του κέρδος είναι πάντα μικρότερο, όπως φαίνεται και στο διάγραμμα 7.1, από τον ορισμό της απόδοσης προκύπτει ότι ο incPCA θα έχει μικρότερο σφάλμα. Ο PC3 έχει οριακά καλύτερη απόδοση από τους infHorPC31 και infHorPC32, επειδή όμως ο PC3 έχει πάντα μικρότερο ενεργειακό κέρδος οι άλλοι δύο αλγόριθμοι θα έχουν μεγαλύτερο σφάλμα.



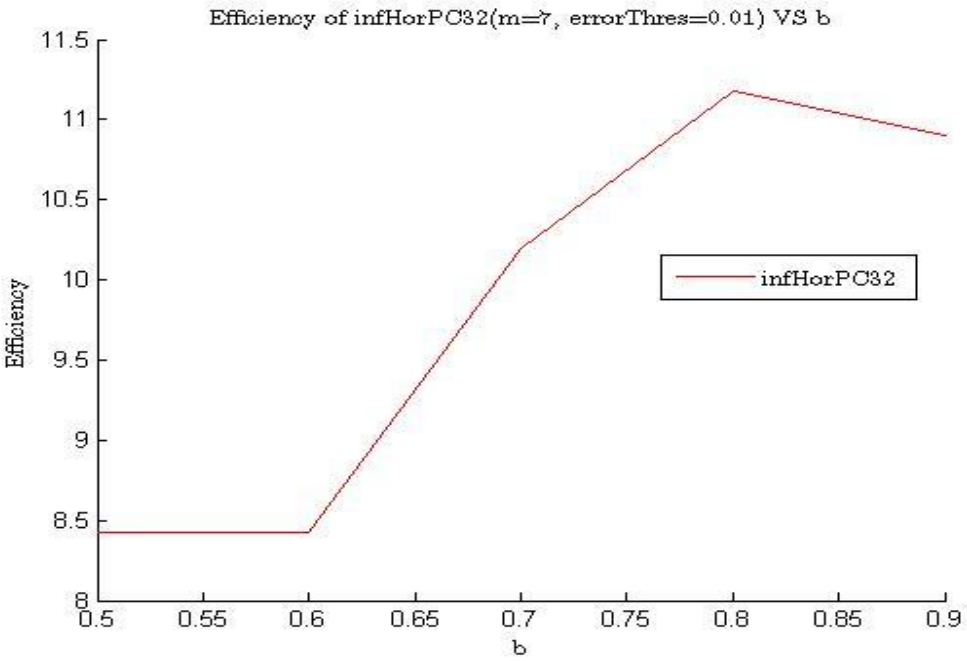
Διάγραμμα 7.12 Απόδοση των τεσσάρων σχημάτων ως προς τον χρόνο με τη δεύτερη μετρική για εκτίμηση της απόδοσης, για μήκος φάσης εκμάθησης m=7 και κατώφλι σφάλματος errorThres=1%.

Επειδή το διάγραμμα 7.11 της απόδοσης των τεσσάρων αλγορίθμων με την πρώτη μετρική είναι πιθανό να δημιουργεί τη στρεβλή εικόνα ότι ο PC3 είναι «καλύτερος» των infHorPC31 και infHorPC32, χρησιμοποιήσαμε στο σημείο αυτό τη δεύτερη μετρική για την απόδοση New Efficiency Metric. Βάσει αυτής, όπως βλέπουμε στο διάγραμμα 7.12, μπορούμε να κατανοήσουμε ότι η αύξηση στο σχετικό σφάλμα που παράγεται από τους infHorPC31 και infHorPC32 δεν είναι τέτοια ώστε να επισκιάζει το μεγαλύτερο ενεργειακό κέρδος που επιτυγχάνουν έναντι του PC3, εφόσον όταν το ενεργειακό κέρδος και η πιστότητα στα δεδομένα έχουν την ίδια βαρύτητα στην εκτίμηση της επίδοσης, τα σχήματα μη πεπερασμένου ορίζοντα συμπίεσης φαίνεται να υπερτερούν.



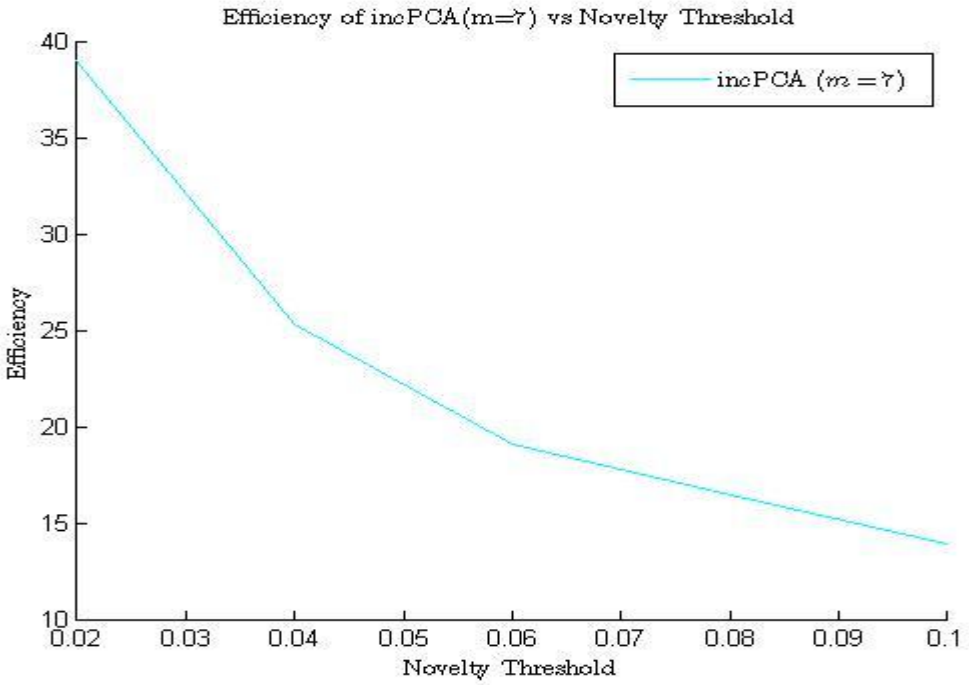
Διάγραμμα 7. 13 Μεταβολή της απόδοσης ως προς το μήκος φάσης εκμάθησης για όλους τους αλγορίθμους.

Μεταβάλλοντας την τιμή του ορίζοντα εκμάθησης m και συγκρίνοντας τους αλγορίθμους ως προς την απόδοση (διάγραμμα 7.13), διαπιστώνουμε ότι για όλες τις τιμές που παίρνει το m ο incPCA είναι πιο αποδοτικός από τους άλλους, με εξαίρεση την τιμή του ορίζοντα εκμάθησης m=5 για την οποία η απόδοση του incPCA πέφτει κατακόρυφα. Δεν μπορούμε να εξάγουμε κάποιο ασφαλές συμπέρασμα για την αιτία που προκαλεί αυτή τη μείωση στην απόδοση. Πιθανότατα οφείλεται στη φύση των πειραματικών δεδομένων. Για m=5, εκτός από την μεγάλη πτώση της απόδοσης του incPCA, ενδιαφέρον παρουσιάζει και το γεγονός ότι η απόδοση του infHorPC32 είναι καλύτερη από εκείνη του PC3, ο οποίος για όλες τις άλλες τιμές της φάσης εκμάθησης είναι καλύτερος, σε ότι αφορά την απόδοση, και από τον infHorPC32 και infHorPC31.



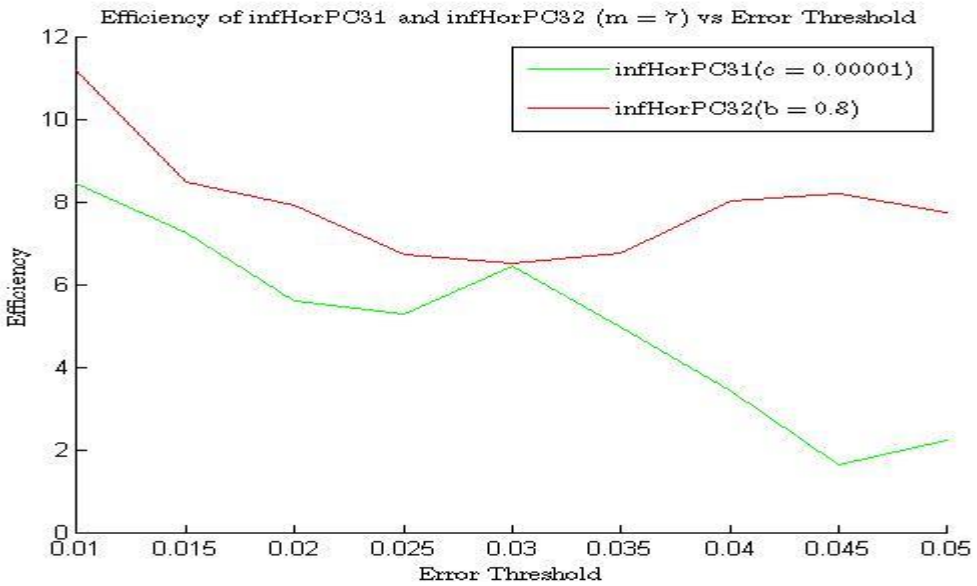
Διάγραμμα 7. 14 Μεταβολή της απόδοσης ως προς την πιθανότητα κέρδους για μήκος φάσης εκμάθησης m=7 και κατώφλι σφάλματος errorThres=1% για τον αλγόριθμο infHorPC32.

Στο παραπάνω διάγραμμα παρατηρούμε πως μεταβάλλεται η απόδοση του αλγορίθμου infHorPC32 για τις διαφορετικές τιμές της πιθανότητας κέρδους b και για σταθερό m=5 και errorThres=0.01. Παρατηρούμε ότι για τιμές πιθανότητας μικρότερες του 0.65, η απόδοση παραμένει σταθερή. Επομένως, δεν έχει νόημα να τρέξουμε τον αλγόριθμο για μικρότερες τιμές. Εξάλλου, για b=0.5 η πιθανότητα κέρδους είναι ίδια με την πιθανότητα απώλειας. Για τιμές πιθανότητας από 0.65 και πάνω, η απόδοση του infHorPC32 αρχίζει να αυξάνεται και λαμβάνει την μέγιστη τιμή της για b=0.8. Για b μεγαλύτερη από 0.8, η απόδοση μειώνεται. Αυτή η μείωση της απόδοσης οφείλεται στο γεγονός ότι η πιθανότητα να έχουμε κέρδος είναι πολύ μεγάλη, άρα ο αλγόριθμος θα κάνει περισσότερη συμπίεση με αποτέλεσμα να αυξάνεται το σφάλμα.



Διάγραμμα 7. 15 Μεταβολή της απόδοσης ως προς το κατώφλι καινοτομίας για μήκος φάσης εκμάθησης m=7 για τον αλγόριθμο incPCA.

Η απόδοση του incPCA μειώνεται όσο αυξάνεται η τιμή του novelty threshold, όπως μας δείχνει το διάγραμμα 7.15. Ο λόγος που συμβαίνει αυτό είναι ότι όσο αυξάνεται η τιμή του novelty threshold, ο αλγόριθμος γίνεται πιο ανεκτικός στα σφάλματα με αποτέλεσμα να μεγαλώνει το μέσο σχετικό σφάλμα, άρα σύμφωνα με τον ορισμό της, η απόδοση θα μειώνεται. Επίσης, βλέπουμε ότι ο incPCA λαμβάνει τη μέγιστη τιμή απόδοσης για τιμή novelty threshold 0.02, για αυτό στις συγκρίσεις που κάνουμε ανάμεσα στον incPCA και στους άλλους αλγορίθμους θέτουμε το novelty threshold ίσο με 0.02

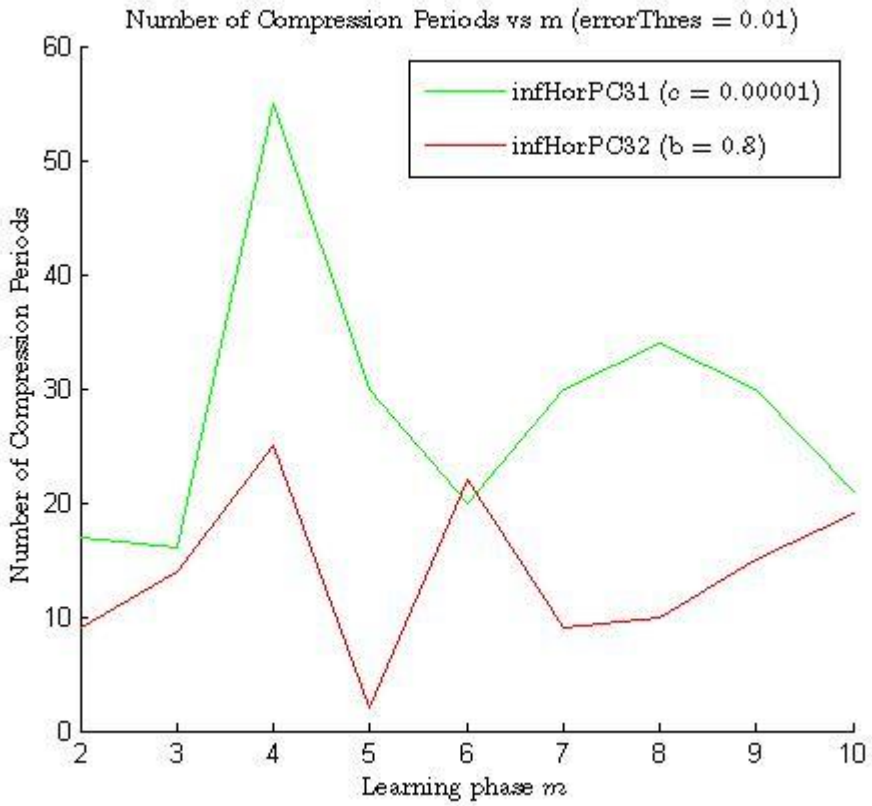


Διάγραμμα 7. 16 Μεταβολή της απόδοσης ως προς το κατώφλι σφάλματος για μήκος φάσης εκμάθησης m=7 για τους αλγορίθμους infHorPC31 και infHorPC32.

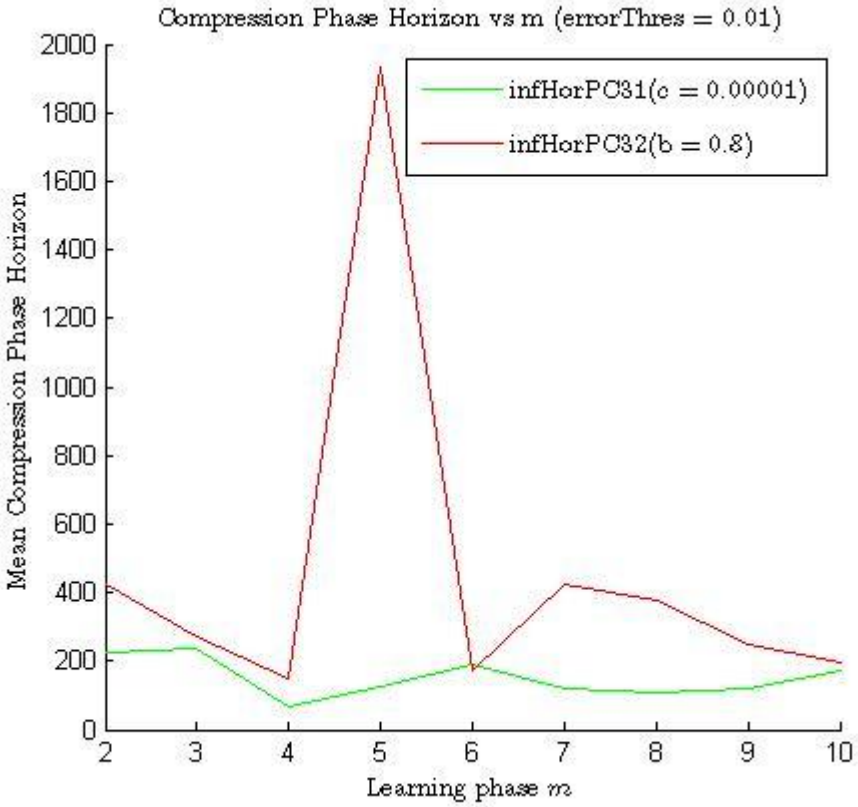
Τέλος, μεταβάλλοντας την τιμή του error threshold συγκρίναμε τους αλγορίθμους infHorPC31 και infHorPC32 ως προς την απόδοση. Τα συμπεράσματα στα οποία

καταλήξαμε είναι ότι για όλες τις τιμές του error threshold ο infHorPC32 είναι αποδοτικότερος από τον infHorPC31 και ότι και οι δύο αλγόριθμοι λαμβάνουν τη μέγιστη τιμή απόδοσης για errorThres=0.01. Αυτός είναι και ο λόγος για τον οποίο στις συγκρίσεις που κάνουμε μεταξύ των αλγορίθμων επιλέξαμε το error threshold να έχει τιμή 0.01.

7.4.4 Αριθμός φάσεων συμπίεσης και μέσο μήκος των φάσεων συμπίεσης



Διάγραμμα 7. 17 Μεταβολή του πλήθους των περιόδων συμπίεσης ως προς το μήκος της φάσης εκμάθησης για τιμή κατωφλιού σφάλματος 1% για τους αλγορίθμους infHorPC31 και infHorPC32.



Διάγραμμα 7.18 Μεταβολή μέσου μήκους περιόδου συμπίεσης ως προς το μήκος της φάσης εκμάθησης για τιμή κατωφλιού σφάλματος 1% για τους αλγορίθμους infHorPC31 και infHorPC32.

Στα παραπάνω διαγράμματα συγκρίνουμε τους infHorPC31 και infHorPC32 ως προς τον αριθμό των φάσεων συμπίεσης και το μέσο μήκος των φάσεων συμπίεσης για διαφορετικές τιμές του μήκους της φάσης εκμάθησης. Όπως βλέπουμε στο διάγραμμα 7.17, ο infHorPC32 έχει λιγότερες περιόδους συμπίεσης από τον infHorPC31. Επομένως, δικαιολογείται το γεγονός ότι ο infHorPC32 έχει μεγαλύτερο μήκος συμπίεσης, διάγραμμα 7.18, όπως επίσης και το γεγονός ότι έχει μεγαλύτερο σφάλμα και λιγότερη κατανάλωση ενέργειας, όπως είδαμε νωρίτερα.

ΕΠΙΛΟΓΟΣ

Συνοψίζοντας, στην παρούσα διπλωματική παρουσιάσαμε τρία νέα σχήματα για την συμπίεση δεδομένων σε ένα δίκτυο αισθητήρων. Τα σχήματα αυτά βασίστηκαν στον αλγόριθμο PC3. Όπως είδαμε, τα δυο εξ αυτών των σχημάτων, ο infHorPC31 και ο infHorPC32, χρησιμοποιούν τη θεωρία βέλτιστης παύσης για τον καθορισμό της συνθήκης τερματισμού της φάσης συμπίεσης, ενώ ο incPCA χρησιμοποιεί κατά τη συμπίεση τη μέθοδο incremental PCA. Τα αποτελέσματα των πειραμάτων έδειξαν ότι οι αλγόριθμοι infHorPC31 και infHorPC32 είναι καλύτεροι ως προς την κατανάλωση ενέργειας συγκρινόμενοι με τον PC3. Έχουν, όμως, μεγαλύτερο σφάλμα από αυτόν. Ο incPCA έχει το μικρότερο ενεργειακό κέρδος από όλους. Πολλές φορές, μάλιστα, η ενέργεια που καταναλώνει είναι μεγαλύτερη από εκείνη που θα καταναλώνονταν αν δεν κάναμε καθόλου συμπίεση. Ωστόσο, ο incPCA επιτυγχάνει το μικρότερο σφάλμα από όλους. Τα αποτελέσματα αυτά, βέβαια, έχουν να κάνουν με την επιλογή των τιμών των παραμέτρων και μπορεί να διαφοροποιηθούν αρκετά για διαφορετικές τιμές στις παραμέτρους. Ακόμη, εξαρτώνται σημαντικά από το πειραματικό σύνολο, το οποίο στην περίπτωσή μας είναι αρκετά συσχετισμένο. Για ένα λιγότερο συσχετισμένο πειραματικό σύνολο πιθανότατα όλοι οι αλγόριθμοι θα είχαν μεγαλύτερη κατανάλωση ενέργειας. Η επιλογή του αλγορίθμου που θα χρησιμοποιήσουμε εξαρτάται από το στόχο της εφαρμογής μας, δηλαδή, αν θέλουμε να πετύχουμε μικρή ενέργεια ή μικρό σφάλμα. Βέβαια, όπως είδαμε στην εισαγωγή, σε ένα δίκτυο αισθητήρων μεγαλύτερη σημασία από όλα έχει να διατηρήσουμε την κατανάλωση ενέργειας σε χαμηλά επίπεδα, καθώς η διαθέσιμη ενέργεια των κόμβων – αισθητήρων είναι περιορισμένη. Επομένως, για ένα τέτοιο δίκτυο καλύτερη επιλογή αποτελούν οι αλγόριθμοι infHorPC31 και infHorPC32. Μελλοντικός μας στόχος είναι η βελτίωση των προτεινόμενων αλγορίθμων.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός όρος**
contextual information	Πληροφορία πλαισίου
optimal stopping theory	Θεωρία βέλτιστης παύσης
incremental principal component analysis	Αυξητική ανάλυση κύριων συνιστωσών
Source	Πηγή
Relay	αναμεταδότης
Context vectors	Διανύσματα πλαισίου
Sender node	Κόμβος- αποστολέας
Receiver node	Κόμβος-παραλήπτης
principal component analysis	Ανάλυση κύριων συνιστωσών
residual vector	Υπολειπόμενο διάνυσμα
principal components	Κύριες συνιστώσες
learning phase	Φάση εκμάθησης
(de)compression phase	Φάση (απο)συμπίεσης
backward induction	Ανάδρομη επαγωγή
Accuracy	Ακρίβεια
discounted sum	Μειούμενο άθροισμα
the burglar problem	Το πρόβλημα του διαρρήκτη
error threshold	Κατώφλι σφάλματος
Kernel Density Estimator	Εκτιμητής πυκνότητας Kernel
probability density function	Συνάρτηση πυκνότητας πιθανότητας
cumulative distribution function	Συσσωρευτική συνάρτηση κατανομής
error function	Συνάρτηση σφάλματος
residue vector	Υπολειπόμενο διάνυσμα
Novelty threshold	Παράγοντας καινοτομίας
Energy Consumption	Κατανάλωση ενέργειας
Energy gain	Ενεργειακό κέρδος
Efficiency	απόδοση
Mean total relative error	Μέσο συνολικό σχετικό σφάλμα
Compression ratio	Ποσοστό συμπίεσης

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ΜΗΜΣ	μικρο-ηλεκτρομηχανικά συστήματα
PC3	Principal component–based context compression
OST	optimal stopping theory
CVs	Context vectors
PCA	Principal component analysis
PCs	Principal components
KDE	Kernel Density Estimator
PDF	probability density function
CDF	cumulative distribution function
mtre	Mean total relative error

ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα αυτό δίνουμε τους κώδικες των αλγορίθμων των σχημάτων PC3, infHorPC31, infHorPC32 και incPCA. Σημειώνουμε και πάλι ότι η υλοποίηση έγινε στο περιβάλλον MatlabR2009b. Στα κυριότερα σημεία των κωδίκων υπάρχουν σχόλια στην αγγλική γλώσσα.

PC3:

Σημειώνουμε ότι ο αλγόριθμος αυτός παίρνει τις εξής τρεις εισόδους:

- τον ορίζοντα της φάσης εκμάθησης m,
- τον ορίζοντα της φάσης συμπίεσης l, και τέλος
- την ακρίβεια accur που καθορίζει τον αριθμό των κύριων συνιστωσών.

Στην έξοδο δίνει τα ακόλουθα:

- τον πίνακα Energy με τη συνολική ενέργεια που καταναλώνεται μέχρι ένα βήμα (ο πίνακας αυτός έχει μέγεθος ίσο με το μέγεθος των δειγμάτων του πειραματικού συνόλου δεδομένων μας),
- τον πίνακα C με το μέσο σχετικό σφάλμα που παράγεται για κάθε δείγμα,
- το ποσό της συνολικής κατανάλωσης ενέργειας totalEnergyConsumption σε nJoule,
- το μέσο συνολικό σχετικό σφάλμα meanTotalRelError,
- τον πίνακα Q που κρατά τον αριθμό των κύριων συνιστωσών που επιλέγονται μετά από κάθε φάση εκμάθησης,
- τον πίνακα SendDataPlot με το σύνολο των δεδομένων που έχουν αποσταλεί μέχρι ένα ορισμένο βήμα,
- τον πίνακα UncEnergyPerSample με την συνολική ενέργεια που θα καταναλωνόταν αν δεν είχαμε συμπίεσης μέχρι ένα ορισμένο βήμα,
- το ποσοστό του ενεργειακού κέρδους energyGain, και
- τον πίνακα με ποσό συμπίεσης xCompression που επιτυγχάνει συνολικά μέχρι κάποιο βήμα.

Ακολουθεί η συνάρτηση.

```
function [ Energy, C, totalEnergyConsumption, meanTotalRelError, Q, SendDataPlot, UncEnergyPerSample, energyGain, xCompression ] = PC3( m, l, accur)


%load data

load clear_sensor_values.txt;

data = clear_sensor_values;

data = [data; data; data; data; data; data; data; data; data; data;];

data(388:750,:) = data(388:750,:) + 0.0975*10;

data(751:1200, :) = flipud(data(751:1200, :));

data(2100:2499, :) = 1.1 * data(3100:3499, :);


%variables' initialization

endOfData = 0;

curSample = 0;
```

```
dataSize = size(data,1);
horLength = [];
energyPerSample = 0;
n = 7;
Energy = [];
SendDataPlot = [];
totalEnergyConsumption = 0;
totalUncomprEnergy = 0;
UncEnergyPerSample = [];
meanTotalRelError = 0;
C = [];
Q=[];
xCompression = [];
sendData = 0;
sendData_without_PC3 = 0;


energy_needed_per_transmitted_bit  = 720; %nJoules/bit
energy_needed_per_received_bit  = 110;% nJoules/bit
energy_needed_per_operation = 4; %nJoule/operation


while(endOfData == 0)


    %learning phase
    remainData = dataSize - curSample;
    %energyPerSample = energy needed for the transmission and the reception
of each of learning phase samples(uncompressed vectors)
    energyPerSample                    =                    transmitValuesCost(n,
energy_needed_per_transmitted_bit)           +           receiveValuesCost(n,
energy_needed_per_received_bit);
    if(remainData >= m)
        x = data((curSample+1):(curSample+m),:);
        %we send each of m-1 samples separately, then we send the mth
        %sample and we compute the base.
        for i=1:1:m;
            C = [C 0];
            SendDataPlot = [SendDataPlot n];
            if(i == m)
                % add to totalEnergyConsumption the energy needed for the
transmission and
                % the reception of the m-th sample and the energy needed for
the creation of the base
                totalEnergyConsumption     =     totalEnergyConsumption     +
energyPerSample + createPcaCoeffCost(n,m,energy_needed_per_operation);
```

```
            else
                totalEnergyConsumption = totalEnergyConsumption + energyPerSample;
            end
            Energy = [Energy totalEnergyConsumption];
            totalUncomprEnergy = totalUncomprEnergy + energyPerSample;
            UncEnergyPerSample = [UncEnergyPerSample (totalUncomprEnergy)];
            sendData = sendData + n;
            sendData_without_PC3 = sendData_without_PC3 + n;
            xCompression = [xCompression sendData/sendData_without_PC3];
        end
    else
        endOfData = 1;
        for i=1:1:remainData;
            C = [C 0];
            SendDataPlot = [SendDataPlot n];
            totalEnergyConsumption = totalEnergyConsumption + energyPerSample;
            Energy = [Energy totalEnergyConsumption];
            totalUncomprEnergy = totalUncomprEnergy + energyPerSample;
            UncEnergyPerSample = [UncEnergyPerSample totalUncomprEnergy];
            sendData = sendData + n;
            sendData_without_PC3 = sendData_without_PC3 + n;
            xCompression = [xCompression sendData/sendData_without_PC3];
        end
        break;
    end
    %getting principal components
    [base,score,lamda] = princomp(x);
    %determine the first q PCs that describe the entire matrix X with accuracy
    %accur%
    q = 7;
    sumall = sum(lamda);
    Sum = 0;
    for i=1:length(lamda)
        Sum = Sum + lamda(i)/sumall;
        if(Sum >= accur)
            q = i;
            break;
        end
    end
```

```
    %Q = current number of PCs
    Q = [Q q];
    baseq = base(:,1:q);
    Mean = mean(x,1);
    mx = Mean';
    curSample = curSample + m;
    k = 0; %k is the compression  step
    while(k < l && endOfData == 0)
        %let the new vector come = y
        if(curSample == dataSize)
            endOfData = 2;
            break;
        end
        curSample = curSample + 1;
        k = k + 1;
        y = data(curSample,:)';
        %g = project of y onto baseq
        g = baseq'*(y-mx);
        ynew = baseq*g + mx;
        %h = residue vector
        h = y - ynew;
        relError = norm(h)/norm(y);
        C = [C relError];
        if(k == 1) %first compression step
            %add to totalEnergyConsumption the energy needed for the
            %vector's compression, the energy for the tranmission
            %and reception of the compressed vector and of the base, and the
            %energy for the vector's uncompression
            totalEnergyConsumption      =      totalEnergyConsumption      +
PCsCompressCost(n,q,energy_needed_per_operation)+
transmitValuesCost(q+n*q,energy_needed_per_transmitted_bit)                +
receiveValuesCost(q+n*q,energy_needed_per_received_bit)                    +
PCsUncompressCost(n,q,energy_needed_per_operation);
            totalUncomprEnergy = totalUncomprEnergy + energyPerSample;
            SendDataPlot = [SendDataPlot n*q + q];
            sendData = sendData + n*q + q;
            sendData_without_PC3 = sendData_without_PC3 + n;
            xCompression = [xCompression sendData/sendData_without_PC3];
        else
            %add to totalEnergyConsumption the energy needed for the
            %vector's compression, the energy for the vector's transmission
            %and reception, and the energy for the vector's uncompression
```

```
            totalEnergyConsumption      =      totalEnergyConsumption      +
PCsCompressCost(n,q,energy_needed_per_operation)+
transmitValuesCost(q,energy_needed_per_transmitted_bit)                    +
receiveValuesCost(q,energy_needed_per_received_bit)                        +
PCsUncompressCost(n,q,energy_needed_per_operation);
            totalUncomprEnergy = totalUncomprEnergy + energyPerSample;
            SendDataPlot = [SendDataPlot q];
            sendData = sendData + q;
            sendData_without_PC3 = sendData_without_PC3 + n;
            xCompression = [xCompression sendData/sendData_without_PC3];
        end
        Energy = [Energy totalEnergyConsumption];
        UncEnergyPerSample = [UncEnergyPerSample totalUncomprEnergy];
    end
end
    meanTotalRelError = sum(C)/dataSize;
    cost_without_PC3               =               dataSize               *
(transmitValuesCost(n,energy_needed_per_transmitted_bit)                   +
receiveValuesCost(n,energy_needed_per_received_bit));
    energyGain               =               (cost_without_PC3             -
totalEnergyConsumption)/cost_without_PC3;
end
```

infHorPC31:

Σημειώνουμε ότι ο αλγόριθμος αυτός παίρνει τις εξής τέσσερις εισόδους:

- τον ορίζοντα της φάσης εκμάθησης m,
- το κατώφλι σφάλματος errorThres,
- το κόστος για κάθε νέο δείγμα c, και τέλος
- την ακρίβεια accur βάσει της οποίας υπολογίζεται ο αριθμός των κύριων συνιστωσών στον πίνακα συμπίεσης.

Στην έξοδο δίνει τα ακόλουθα:

- τον πίνακα Energy με τη συνολική ενέργεια που καταναλώνεται μέχρι ένα βήμα (ο πίνακας αυτός έχει μέγεθος ίσο με το μέγεθος των δειγμάτων του πειραματικού συνόλου δεδομένων μας),
- τον πίνακα C με το μέσο σχετικό σφάλμα που παράγεται για κάθε δείγμα, το ποσό της συνολικής κατανάλωσης ενέργειας σε nJoule,
- τον πίνακα horLength με το μήκος του ορίζοντα διάδοσης για κάθε φάση συμπίεσης (ο πίνακας αυτός θα έχει μέγεθος τόσο όσες είναι οι φάσεις συμπίεσης),
- το συνολικό ποσό της ενέργειας που σπαταλάμε totalEnergyConsumption με το σχήμα αυτό σε nJoule,
- το μέσο συνολικό σχετικό σφάλμα meanTotalRelError,
- τον πίνακα Q που κρατά τον αριθμό των κύριων συνιστωσών που επιλέγονται μετά από κάθε φάση εκμάθησης,

- τον πίνακα SendDataPlot με το σύνολο των δεδομένων που έχουν αποσταλεί μέχρι ένα ορισμένο βήμα,
- τον πίνακα UncEnergyPerSample με την συνολική ενέργεια που θα καταναλωνόταν αν δεν είχαμε συμπίεσης μέχρι ένα ορισμένο βήμα,
- το ποσοστό του ενεργειακού κέρδους energyGain, και
- τον πίνακα με ποσό συμπίεσης xCompression που επιτυγχάνει συνολικά μέχρι κάποιο βήμα.

Ακολουθεί η συνάρτηση.

```
function [ Energy, C, horLength, totalEnergyConsumption, meanTotalRelError, Q, SendDataPlot, UncEnergyPerSample, energyGain, xCompression ] = infHorPC31( m, errorThres, c, accur)


%load data
load clear_sensor_values.txt;
data = clear_sensor_values;
data = [data; data; data; data; data; data; data; data; data; data;];
data(388:750,:) = data(388:750,:) + 0.0975*10;
data(751:1200, :) = flipud(data(751:1200, :));
data(2100:2499, :) = 1.1 * data(3100:3499, :);


%variables' initialization
endOfData = 0;
curSample = 0;
dataSize = size(data,1);
horLength = [];
energyPerSample = 0;
n = 7;
Energy = [];
SendDataPlot = [];
totalEnergyConsumption = 0;
totalUncomprEnergy = 0;
UncEnergyPerSample = [];
meanTotalRelError = 0;
C = [];
Q=[];
xCompression = [];
sendData = 0;
sendData_without_infHorPC31 = 0;


energy_needed_per_transmitted_bit  = 720; %nJoules/bit
energy_needed_per_received_bit  = 110;% nJoules/bit
```

```
energy_needed_per_operation = 4; %nJoule/operation


while(endOfData == 0)


    %learning phase
    remainData = dataSize - curSample;
    %energyPerSample = energy needed for the transmission and the reception
of each of learning phase samples
    energyPerSample                    =                    transmitValuesCost(n,
energy_needed_per_transmitted_bit)           +           receiveValuesCost(n,
energy_needed_per_received_bit);
    if(remainData >= m)
        x = data((curSample+1):(curSample+m),:);
        %we send each of m-1 samples separately, then we send the mth
        %sample and we compute the base of eigenvectors
        for i=1:1:m;
            C = [C 0];
            SendDataPlot = [SendDataPlot n];
            if(i == m)
                % add to totalEnergyConsumption the energy needed for the
transmission and
                % the reception of the m-th sample and the energy needed for
the creation of the base
                totalEnergyConsumption     =     totalEnergyConsumption     +
energyPerSample + createPcaCoeffCost(n,m,energy_needed_per_operation);
            else
                totalEnergyConsumption     =     totalEnergyConsumption     +
energyPerSample;
            end
            Energy = [Energy totalEnergyConsumption];
            totalUncomprEnergy = totalUncomprEnergy + energyPerSample;
            UncEnergyPerSample = [UncEnergyPerSample (totalUncomprEnergy)];
            sendData = sendData + n;
            sendData_without_infHorPC31 = sendData_without_infHorPC31 + n;
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC31];
        end
    else
        endOfData = 1;
        for i=1:1:remainData;
            C = [C 0];
            SendDataPlot = [SendDataPlot n];
            totalEnergyConsumption      =      totalEnergyConsumption      +
energyPerSample;
```

```
            Energy = [Energy totalEnergyConsumption];
            totalUncomprEnergy = totalUncomprEnergy + energyPerSample;
            UncEnergyPerSample = [UncEnergyPerSample (totalUncomprEnergy)];
            sendData = sendData + n;
            sendData_without_infHorPC31 = sendData_without_infHorPC31 + n;
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC31];
        end
        break;
    end
    %getting principal components
    [base,score,lamda] = princomp(x);
    %determine the first q PCs that describe the entire matrix X with
accuracy
    %accur%
    q = 7;
    sumall = sum(lamda);
    Sum = 0;
    for i=1:length(lamda)
        Sum = Sum + lamda(i)/sumall;
        if(Sum >= accur)
            q = i;
            break;
        end
    end
    %Q = current number of PCs
    Q = [Q q];
    baseq = base(:,1:q);
    Mean = mean(x,1);
    mx = Mean';
    curSample = curSample + m;
    currentError = 1.1; % we set the error equal to 1.1 so as to enter the
while loop
    k = 0; %k is the compression  horizon
    while(currentError > sqrt(2*c) && endOfData == 0)
        %let the new vector come = y
        if(curSample == dataSize)
            endOfData = 2;
            break;
        end
        curSample = curSample + 1;
        k = k + 1;
```

```
        y = data(curSample,:)';
        %g = project of y onto baseq
        g = baseq'*(y-mx);
        ynew = baseq*g + mx;
        %h = residue vector
        h = y - ynew;
        relError = norm(h)/norm(y);
        C = [C relError];
        if(relError <= errorThres)
            currentError = 0;
        else
            currentError = min([1 relError]);
        end
        if(k == 1) %first compression step
            %add to totalEnergyConsumption the energy needed for the
            %vector's compression, the energy for the tranmission
            %and reception of the compressed vector and of the base, and the
            %energy for the vector's uncompression
            totalEnergyConsumption      =      totalEnergyConsumption      +
PCsCompressCost(n,q,energy_needed_per_operation)+
transmitValuesCost(q+n*q,energy_needed_per_transmitted_bit)                +
receiveValuesCost(q+n*q,energy_needed_per_received_bit)                    +
PCsUncompressCost(n,q,energy_needed_per_operation);
            SendDataPlot = [SendDataPlot n*q + q];
            sendData = sendData + n*q + q;
            sendData_without_infHorPC31 = sendData_without_infHorPC31 + n;
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC31];


        else
            %add to totalEnergyConsumption the energy needed for the
            %vector's compression, the energy for the vector's tranmission
            %and reception, and the energy for the vector's uncompression
            totalEnergyConsumption      =      totalEnergyConsumption      +
PCsCompressCost(n,q,energy_needed_per_operation)+
transmitValuesCost(q,energy_needed_per_transmitted_bit)                    +
receiveValuesCost(q,energy_needed_per_received_bit)                        +
PCsUncompressCost(n,q,energy_needed_per_operation);
            SendDataPlot = [SendDataPlot q];
            sendData = sendData + q;
            sendData_without_infHorPC31 = sendData_without_infHorPC31 + n;
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC31];
        end
```

```
        Energy = [Energy totalEnergyConsumption];
        totalUncomprEnergy = totalUncomprEnergy + energyPerSample;
        UncEnergyPerSample = [UncEnergyPerSample (totalUncomprEnergy)];
    end
    horLength = [horLength k];
end
    meanTotalRelError = sum(C)/dataSize;
    cost_without_infHorPC31 = dataSize *
(transmitValuesCost(n,energy_needed_per_transmitted_bit) +
receiveValuesCost(n,energy_needed_per_received_bit));
    energyGain = (cost_without_infHorPC31 -
totalEnergyConsumption)/cost_without_infHorPC31;
end
```

infHorPC32:

Σημειώνουμε ότι ο αλγόριθμος αυτός παίρνει τις εξής τέσσερεις εισόδους:

- τον ορίζοντα της φάσης εκμάθησης m,
- το κατώφλι σφάλματος errorThres,
- την πιθανότητα διατήρησης του κέρδους που έχουμε συσσωρεύσει b, και τέλος
- την ακρίβεια accur βάσει της οποίας υπολογίζεται ο αριθμός των κύριων συνιστωσών στον πίνακα συμπίεσης.

Στην έξοδο δίνει τα ακόλουθα:

- τον πίνακα C με το μέσο σχετικό σφάλμα που παράγεται για κάθε δείγμα, το ποσό της συνολικής κατανάλωσης ενέργειας σε nJoule,
- τον πίνακα Energy με τη συνολική ενέργεια που καταναλώνεται μέχρι ένα βήμα (ο πίνακας αυτός έχει μέγεθος ίσο με το μέγεθος των δειγμάτων του πειραματικού συνόλου δεδομένων μας),
- τον πίνακα horLength με το μήκος του ορίζοντα διάδοσης για κάθε φάση συμπίεσης (ο πίνακας αυτός θα έχει μέγεθος τόσο όσες είναι οι φάσεις συμπίεσης),
- το μέσο συνολικό σχετικό σφάλμα meanTotalRelError,
- το συνολικό ποσό της ενέργειας που θα καταναλώνουμε με το σχήμα αυτό totalEnergyConsumption σε nJoule,
- τον πίνακα Q που κρατά τον αριθμό των κύριων συνιστωσών που επιλέγονται μετά από κάθε φάση εκμάθησης,
- τον πίνακα SendDataPlot με το σύνολο των δεδομένων που έχουν αποσταλεί μέχρι ένα ορισμένο βήμα,
- το ποσοστό του ενεργειακού κέρδους energyGain,
- τον πίνακα EnergyGainVector με το συνολικό ενεργειακό κέρδος σε κάθε βήμα, και
- τον πίνακα xCompression με ποσό συμπίεσης που επιτυγχάνει συνολικά μέχρι κάποιο βήμα.

Ακολουθεί η συνάρτηση.

```
function [ C, Energy, horLength, meanTotalRelError, totalEnergyConsumption, Q, SendDataPlot, energyGain, EnergyGainVector, xCompression ] = infHorPC32( m, errorThres, b, accur)

%load data
load clear_sensor_values.txt;
data = clear_sensor_values;
data = [data; data; data; data; data; data; data; data; data; data;];
data(388:750,:) = data(388:750,:) + 0.0975*10;
data(751:1200, :) = flipud(data(751:1200, :));
data(2100:2499, :) = 1.1 * data(3100:3499, :);

%variables' initialization
endOfData = 0;
curSample = 0;
dataSize = size(data,1);
horLength = [];
energyPerSample = 0;
n = 7;
Energy = [];
SendDataPlot = [];
meanTotalRelError = 0;
totalEnergyConsumption = 0;
C = [];
Q=[];
xCompression = [];
EnergyGainVector = [];
sendData = 0;
sendData_without_infHorPC32 = 0;

energy_needed_per_transmitted_bit  = 720; %nJoules/bit
energy_needed_per_received_bit  = 110;% nJoules/bit
energy_needed_per_operation = 4; %nJoule/operation

while(endOfData == 0)

    %learning phase
    remainData = dataSize - curSample;
```

```
    %energyPerSample = energy needed for the transmission and the reception
of each of learning phase samples
    energyPerSample                    =                    transmitValuesCost(n,
energy_needed_per_transmitted_bit)           +           receiveValuesCost(n,
energy_needed_per_received_bit);
    if(remainData >= m)
        x = data((curSample+1):(curSample+m),:);
        %we send each of m-1 samples separately, then we send the mth
        %sample and we compute the base


        for i=1:1:m;
            C = [C 0];
            SendDataPlot = [SendDataPlot n];
            totalEnergyConsumptionWithoutInfHorPC32   =   (curSample   +   i)   *
energyPerSample;
            if(i == m)
                % add to totalEnergyConsumption the energy needed for the
transmission and
                % the reception of the m-th sample and the energy needed for
the creation of the base
                totalEnergyConsumption     =     totalEnergyConsumption     +
energyPerSample + createPcaCoeffCost(n,m,energy_needed_per_operation);
            else
                totalEnergyConsumption     =     totalEnergyConsumption     +
energyPerSample;
            end
            Energy = [Energy totalEnergyConsumption];
            EnergyGainVector                =                [EnergyGainVector
(totalEnergyConsumptionWithoutInfHorPC32                                    -
totalEnergyConsumption)/totalEnergyConsumptionWithoutInfHorPC32];
            sendData = sendData + n;
            sendData_without_infHorPC32 = sendData_without_infHorPC32 + n;
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC32];
        end
    else
        endOfData = 1;
        for i=1:1:remainData;
            C = [C 0];
            SendDataPlot = [SendDataPlot n];
            totalEnergyConsumptionWithoutInfHorPC32   =   (curSample   +   i)   *
energyPerSample;
            totalEnergyConsumption      =      totalEnergyConsumption      +
energyPerSample;
            Energy = [Energy totalEnergyConsumption];
```

```
            EnergyGainVector                =                [EnergyGainVector
(totalEnergyConsumptionWithoutInfHorPC32                                        -
totalEnergyConsumption)/totalEnergyConsumptionWithoutInfHorPC32];
            sendData = sendData + n;
            sendData_without_infHorPC32 = sendData_without_infHorPC32 + n;
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC32];
        end
        break;
    end
    %getting the principal components
    [base,score,lamda] = princomp(x);
    %determine the first q PC that describe the entire matrix X with accuracy
    %accur%
    q = 7;
    sumall = sum(lamda);
    Sum = 0;
    for i=1:length(lamda)
        Sum = Sum + lamda(i)/sumall;
        if(Sum >= accur)
            q = i;
            break;
        end
    end
    %Q = current number of PCs
    Q = [Q q];
    baseq = base(:,1:q);
    Mean = mean(x,1);
    mx = Mean';
    curSample = curSample + m;
    totalGain = 0;
    k=0;
    relErrorVector = [];
    stopFactor = 0.1;
    while(totalGain < stopFactor && endOfData == 0)
        totalEnergyConsumptionWithoutInfHorPC32     =          curSample     *
energyPerSample;
        %let the new vector come = y
        if(curSample == dataSize)
            endOfData = 2;
            break;
        end
```

```
        curSample = curSample + 1;
        k = k + 1;
        y = data(curSample,:)';
        %g = project of y onto baseq
        g = baseq'*(y-mx);
        ynew = baseq*g + mx;
        %h = residue vector
        h = y - ynew;
        relError = norm(h)/norm(y);
        C = [C relError];
        relErrorVector = [relErrorVector relError];
        if(relError <= errorThres)
            totalGain = totalGain + 1;
        end
        p = cdf(errorThres, relErrorVector);
        stopFactor = (b/(1-b)) * p;
        if(k == 1) %first compression step
            %add to totalEnergyConsumption the energy needed for the
            %vector's compression, the energy for the tranmission
            %and reception of the compressed vector and the of base, and the
            %energy for the vector's uncompression
            totalEnergyConsumption      =      totalEnergyConsumption      +
PCsCompressCost(n,q,energy_needed_per_operation)+
transmitValuesCost(q+n*q,energy_needed_per_transmitted_bit)                +
receiveValuesCost(q+n*q,energy_needed_per_received_bit)                    +
PCsUncompressCost(n,q,energy_needed_per_operation);
            SendDataPlot = [SendDataPlot n*q + q];
            sendData = sendData + n*q + q;
            sendData_without_infHorPC32 = sendData_without_infHorPC32 + n;
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC32];
        else
            %add to totalEnergyConsumption the energy needed for the
            %vector's compression, the energy for the vector's tranmission
            %and reception, and the energy for the vector's uncompression
            totalEnergyConsumption      =      totalEnergyConsumption      +
PCsCompressCost(n,q,energy_needed_per_operation)+
transmitValuesCost(q,energy_needed_per_transmitted_bit)                    +
receiveValuesCost(q,energy_needed_per_received_bit)                        +
PCsUncompressCost(n,q,energy_needed_per_operation);
            SendDataPlot = [SendDataPlot q];
            sendData = sendData + q;
            sendData_without_infHorPC32 = sendData_without_infHorPC32 + n;
```

```
            xCompression                    =                    [xCompression
sendData/sendData_without_infHorPC32];
        end
        Energy = [Energy totalEnergyConsumption];
        EnergyGainVector                =                [EnergyGainVector
(totalEnergyConsumptionWithoutInfHorPC32                                    -
totalEnergyConsumption)/totalEnergyConsumptionWithoutInfHorPC32];
    end
    horLength = [horLength k];
end
    meanTotalRelError = sum(C)/dataSize;
    cost_without_infHorPC32             =             dataSize             *
(transmitValuesCost(n,energy_needed_per_transmitted_bit)                    +
receiveValuesCost(n,energy_needed_per_received_bit));
    energyGain             =             (cost_without_infHorPC32             -
totalEnergyConsumption)/cost_without_infHorPC32;
end
```

incPCA:

Σημειώνουμε ότι ο αλγόριθμος αυτός παίρνει τις εξής τρεις εισόδους:

- τον ορίζοντα της φάσης εκμάθησης m,
- το κατώφλι καινοτομίας noveltyThres, και τέλος
- την ακρίβεια accur βάσει της οποίας υπολογίζεται ο αριθμός των κύριων συνιστωσών στον πίνακα συμπίεσης.

Στην έξοδο δίνει τα ακόλουθα:

- τον πίνακα Energy με τη συνολική ενέργεια που καταναλώνεται μέχρι ένα βήμα (ο πίνακας αυτός έχει μέγεθος ίσο με το μέγεθος των δειγμάτων του πειραματικού συνόλου δεδομένων μας),
- τον πίνακα C με το μέσο σχετικό σφάλμα που παράγεται για κάθε δείγμα, το ποσό της συνολικής κατανάλωσης ενέργειας σε nJoule,
- τον πίνακα Q που κρατά τον αριθμό των κύριων συνιστωσών που επιλέγονται μετά από κάθε φάση εκμάθησης,
- το συνολικό ποσό της ενέργειας totalEnergyConsumption που θα καταναλώνουμε με το σχήμα αυτό σε nJoule,
- το μέσο συνολικό σχετικό σφάλμα meanTotalRelError,
- τον πίνακα SendDataPlot με το σύνολο των δεδομένων που έχουν αποσταλεί μέχρι ένα ορισμένο βήμα,
- το ποσοστό του ενεργειακού κέρδους energyGain,
- τον πίνακα EnergyGainVector με το συνολικό ενεργειακό κέρδος σε κάθε βήμα, και
- τον πίνακα xCompression με ποσό συμπίεσης που επιτυγχάνει συνολικά μέχρι κάποιο βήμα.

Ακολουθεί η συνάρτηση.

```
function [ Energy, C, Q, totalEnergyConsumption, meanTotalRelError, SendDataPlot, energyGain, EnergyGainVector, xCompression ] = incPCA( m, accur, noveltyThres)


%load data
load clear_sensor_values.txt;
data = clear_sensor_values;
data = [data; data; data; data; data; data; data; data; data; data;];
data(388:750,:) = data(388:750,:) + 0.0975*10;
data(751:1200, :) = flipud(data(751:1200, :));
data(2100:2499, :) = 1.1 * data(3100:3499, :);


dataSize = size(data,1);
dbDistance = [0]; %distance between the old and new base
n = 7;
novelty = 0;
C = [];
numberOfDetection = 0;
energyPerSample = 0;
compressedSampleEnergy = 0;
uncompressedSampleEnergy = 0;
Energy = [];
SendDataPlot = [];
totalEnergyConsumption = 0;
meanTotalRelError = 0;
xCompression = [];
EnergyGainVector = [];
sendData = 0;
sendData_without_incPCA = 0;


energy_per_transmitted_bit  = 720; %nJoules/bit
energy_per_received_bit  = 110;% nJoules/bit
energy_per_operation = 4; %nJoule/operation


%learning phase
x = data(1:m,:);
%we send m samples uncompressed, so we do not include costs for
%compression/uncompression, but only the transmission and reception costs
energyPerSample = transmitValuesCost(n, energy_per_transmitted_bit) + receiveValuesCost(n, energy_per_received_bit);
```

```
for i=1:1:m;
    C = [C 0];
    SendDataPlot = [SendDataPlot n];
    totalEnergyConsumptionWithoutIncPCA = i * energyPerSample;
    totalEnergyConsumption = totalEnergyConsumption + energyPerSample;
    if( i < m )
        %for the first m-1 samples the energy is equal to the energy needed
for the transmissions and the receptions up to a step
        Energy = [Energy totalEnergyConsumption];
    else
        %for the m-th sample we add in the total energy, the energy needed
for the construction of our
        %base of principal components
        totalEnergyConsumption       =       totalEnergyConsumption       +
createPcaCoeffCost(n, m, energy_per_operation);
        Energy = [Energy totalEnergyConsumption];
    end
    EnergyGainVector = [EnergyGainVector (totalEnergyConsumptionWithoutIncPCA
- totalEnergyConsumption)/totalEnergyConsumptionWithoutIncPCA];
    sendData = sendData + n;
    sendData_without_incPCA = sendData_without_incPCA + n;
    xCompression = [xCompression sendData/sendData_without_incPCA];
end


%getting principal components
[base,score,lamda] = princomp(x);
%determine the first q PC that describe the entire matrix X with accuracy
%accur%


q = 7;
sumall = sum(lamda);
Sum = 0;
for i=1:length(lamda)
    Sum = Sum + lamda(i)/sumall;
    if(Sum >= accur)
        q = i;
        break;
    end
end


%Q = current number of PCs
```

```
Q = [q];
baseq = base(:,1:q);
Mean = mean(x,1);
mx = Mean';


for curSample = m + 1:dataSize;
    totalEnergyConsumptionWithoutIncPCA = curSample * energyPerSample;
    %let the new vector come = y
    y = data(curSample,:)';
    %g = project of y onto baseq
    g = baseq'*(y-mx);
    ynew = baseq*g + mx;
    %h = residue vector
    h = y - ynew;
    hnorm = norm(h);
    if(hnorm == 0)
        hhat = zeros(n,1);
    else
        hhat = h/hnorm;
    end
    relError = hnorm/norm(y);
    if(relError > noveltyThres)
        %new mean value
        newmx = (1/curSample)*((curSample-1)*mx + y);
        %calculate gamma
        gamma = hhat'*(y - mx);
        %creation of the diagonial matrix of eigenvalues
        princLamda = lamda(1:q,:);
        L = diag(princLamda);
        %calculate matrix D
        D = ((curSample - 1)/curSample)*[L zeros(q, 1); zeros(1,q) 0] +
((curSample - 1)/curSample^2)*[g*g' gamma*g; gamma*g' gamma^2];
        [R, Lprime] = eig(D);
        %creation of the new base
        newBase = [baseq hhat]*R;
        %we add in total energy the enegy required for the computation of
        %the new eigenvectors
        totalEnergyConsumption = totalEnergyConsumption +
(q+1)^2*energy_per_operation;
        %creation of the vector of eigenvalues
        LprimeVector = diag(Lprime);
        %sort eigenvalues
```

```
        [LprimeVector index] = sort(LprimeVector, 'descend');
        %recompute the number of requred PCs
        newq = 7;
        sumall = sum(LprimeVector);
        Sum = 0;
        for i=1:length(LprimeVector)
            Sum = Sum + LprimeVector(i)/sumall;
            if(Sum >= accur)
                newq = i;
                break;
            end
        end
        newBase = newBase(:,index(1:newq));
        %compare old DB with new DB
        minq = min(q,newq);
        dist = 0;
        for i = 1:minq
            dist = dist + abs(norm(abs(baseq(:,i)) - abs(newBase(:,i))))/norm(baseq(:,i));
        end
        dbDistance = [dbDistance dist];
        if( dist > noveltyThres && newq >= q  )
            %we send the uncompressed vector-sample that has relative reconstruction error greater than the
            %noveltyThres, along with the whole base of eigenvector ( if
            %the (m+1)th sample has reconstruction error more than the
            %noveltyThres and the distance between the two bases is also more
            %than noveltyThres, we act as we would do with every sample,
            %that is we send the uncompressed vector with the new base,
            %since there is no need for the receiver to have the old base )
            uncompressedSampleEnergy = transmitValuesCost(n + newq*n, energy_per_transmitted_bit) + receiveValuesCost(n + newq*n, energy_per_received_bit);
            totalEnergyConsumption = totalEnergyConsumption + uncompressedSampleEnergy;
            Energy = [Energy totalEnergyConsumption];
            EnergyGainVector = [EnergyGainVector (totalEnergyConsumptionWithoutIncPCA - totalEnergyConsumption)/totalEnergyConsumptionWithoutIncPCA];
            SendDataPlot = [SendDataPlot (n + newq*n)];
            sendData = sendData + (n + newq*n);
            sendData_without_incPCA = sendData_without_incPCA + n;
            xCompression = [xCompression sendData/sendData_without_incPCA];
```

```
        else
            if( dist < noveltyThres && newq > q )
                %in case the current sample is the (m+1)th, we send the old
                %base (since we have not send it before) along with the
                %uncompressed vector-sample and the new principal component,
which
                %has n values
                %In any other case, we send the uncompressed vector-sample
that has absolute error greater than the
                %noveltyThres, along with only the (q+1)th eigenvector, which
contains n values (the
                %rest q vectors remain unchanged and we already have them
                %from our old base)
                if (curSample == (m + 1))
                    uncompressedSampleEnergy = transmitValuesCost(n*q + n +
n,   energy_per_transmitted_bit)   +   receiveValuesCost(n*q   +   n   +   n,
energy_per_received_bit);
                    SendDataPlot = [SendDataPlot (n*q + n + n)];
                    sendData = sendData + (n*q + n + n);
                else
                    uncompressedSampleEnergy   =   transmitValuesCost(n   +   n,
energy_per_transmitted_bit)       +       receiveValuesCost(n       +       n,
energy_per_received_bit);
                    SendDataPlot = [SendDataPlot (n + n)];
                    sendData = sendData + (n + n);
                end
                sendData_without_incPCA = sendData_without_incPCA + n;
                xCompression                    =                    [xCompression
sendData/sendData_without_incPCA];
                totalEnergyConsumption     =     totalEnergyConsumption     +
uncompressedSampleEnergy;
                Energy = [Energy totalEnergyConsumption];
                EnergyGainVector              =              [EnergyGainVector
(totalEnergyConsumptionWithoutIncPCA                                          -
totalEnergyConsumption)/totalEnergyConsumptionWithoutIncPCA];


            else
                %otherwise, in case the current sample is the (m+1)th and
since the old base is adequate and we only send the
                %uncompressed vector along with the base that has not been
                %sent and received before
                %In any other case, we send only the uncompressed vector
                %(the base has already been sent and received)
                if (curSample == (m + 1))
```

```
                    uncompressedSampleEnergy = transmitValuesCost(n*q + n,
energy_per_transmitted_bit)      +      receiveValuesCost(n*q      +      n,
energy_per_received_bit);
                    SendDataPlot = [SendDataPlot (n*q + n)];
                    sendData = sendData + (n*q + n);
                else
                    uncompressedSampleEnergy     =     transmitValuesCost(n,
energy_per_transmitted_bit) + receiveValuesCost(n, energy_per_received_bit);
                    SendDataPlot = [SendDataPlot n];
                    sendData = sendData + n;
                end
                sendData_without_incPCA = sendData_without_incPCA + n;
                xCompression                  =                  [xCompression
sendData/sendData_without_incPCA];
                totalEnergyConsumption     =     totalEnergyConsumption     +
uncompressedSampleEnergy;
                Energy = [Energy totalEnergyConsumption];
                EnergyGainVector             =             [EnergyGainVector
(totalEnergyConsumptionWithoutIncPCA                                        -
totalEnergyConsumption)/totalEnergyConsumptionWithoutIncPCA];
            end
        end
        %update
        baseq = newBase(:,1:newq);
        q = newq;
        mx = newmx;
        novelty = 1;
        [q novelty];
        Q = [Q q];
        C = [C 0];
        numberOfDetection = numberOfDetection + 1;
    else
        consumption = 0;
        novelty = 0;
        [q novelty];
        Q = [Q q];
        %in case the current sample is the (m+1)th, we send the base along
        %with the compressed sample, which will now contain q values
        %In any other case, we only send the compressedSample
        if (curSample == (m+1))
            compressedSampleEnergy       =       PCsCompressCost(n,       q,
energy_per_operation)       +       transmitValuesCost(q       +       n*q,
energy_per_transmitted_bit)      +      receiveValuesCost(q      +      n*q,
energy_per_received_bit) + PCsUncompressCost(n, q, energy_per_operation);
```

```
            SendDataPlot = [SendDataPlot (q + n*q)];
            sendData = sendData + (q + n*q);
        else
            compressedSampleEnergy = PCsCompressCost(n, q,
energy_per_operation) + transmitValuesCost(q, energy_per_transmitted_bit) +
receiveValuesCost(q, energy_per_received_bit) + PCsUncompressCost(n, q,
energy_per_operation);
            SendDataPlot = [SendDataPlot q];
            sendData = sendData + q;
        end
        totalEnergyConsumption = totalEnergyConsumption +
compressedSampleEnergy;
        Energy = [Energy totalEnergyConsumption];
        EnergyGainVector = [EnergyGainVector
(totalEnergyConsumptionWithoutIncPCA -
totalEnergyConsumption)/totalEnergyConsumptionWithoutIncPCA];
        sendData_without_incPCA = sendData_without_incPCA + n;
        xCompression = [xCompression sendData/sendData_without_incPCA];
        dbDistance = [dbDistance 0];
        C = [C relError];
    end
end
meanTotalRelError = sum(C)/dataSize;
cost_without_incPCA = dataSize *
(transmitValuesCost(n,energy_per_transmitted_bit) +
receiveValuesCost(n,energy_per_received_bit));
energyGain = (cost_without_incPCA -
totalEnergyConsumption)/cost_without_incPCA;
end
```

ΑΝΑΦΟΡΕΣ

[1] C. Anagnostopoulos, S. Hadjiefthymiades, and P. Georgas, PC3: Principal Component-based Context Compression, J Parallel Distrib Comput 72(2):155-170, 2011.

[2] M. Artac, Mobile Localisation with Incremental PCA, in 11th IEEE Mediterranean Electrotechnical Conference, 2002.

[3] Y. S. Chow, H. Robbins, and D. Siegmund, *Great Expectations: The Theory of Optimal Stopping*. Boston: Houghton Mifflin Company, 1971.

[4] T. S. Ferguson, *Optimal Stopping and Applications*.: Mathematics Department, UCLA, 2008.

[5] P. M. Hall, D. Marshall, R. R. Martin, Incremental Eigenanalysis for Classification, in British Machine Vision Conference, 1998.

[6] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, B. Krogh, Energy-efficient surveillance system using wireless sensor networks, In 2nd ACM International Conference on Mobile Systems, Applications, and Services, ACM MobiSys'04, pp. 270–283.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, *Directed diffusion: a scalable and robust communication paradigm for sensor networks*, Proceedings of the ACM Mobi-Com'00, Boston, MA, pp. 56–67, 2000.

[8] L. J. Snell, Applications of martingale system theorems, *Trans. Amer. Math.*, vol. 73, pp. 293-312, 1952.

[9] A. Wald, Sequential Tests of Statistical Hypotheses, *Ann. Math. Statist.*, vol. 16, no. 2, pp. 117-186, 1945.