Εργασία
Ψηφιακών Τηλεπικοινωνιών

Κυριακόπουλος Αριστομένης[1]

Πανεπιστήμιο Πατρών
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής
up1054307@upnet.gr

Περίληψη Σκοπός της εργασίας είναι η δημιουργία συναρτήσεων για την κατανόηση της κωδικοποίησης Huffman, της κωδικοποίησης ΠΤΜ και την μελέτη απόδοσης ομόδηνου ζωνοπερατού συστήματος Μ-ΦΣΚ. Παρουσιάζονται παρακάτω τα αποτελέσματα της προσπάθειας μου.

1 Ερώτημα 1 - Κωδικοποίηση Huffman

Η κωδικοποίηση Huffman αποτελεί μια βέλτιστη κωδικοποίηση διακριτών πηγών και χρησιμοποιείται για την συμπίεση δεδομένων χωρίς χάσιμο πληροφορίας. Παρακάτω παρατίθεται οι δικές μου θεμελιώσεις των 3 βασικών συναρτήσεων.

1.1 Κωδικοποίηση πηγής κώδικα

Huffman Dictionary Η συνάρτηση huffmanDict λαμβάνει ως είσοδο 2 διανύσματα, το 1 με τους αγγλικούς χαρακτήρες και το άλλο με την πιθανότητα του κάθε γράμματος από το αρχείο cvxopt.txt. Αρχικά ταξινομούνται όλες οι πιθανότητες σε αύξουσα σειρά και λαμβάνονται οι δύο χαρακτήρες με την μικρότερη πιθανότητα να εμφανιστούν. Αυτοί εισέρχονται σε ένα κοινό κόμβο με το άθροισμα των πιθανοτήτων τους και εισέρχονται ξανά στα αντίστοιχα διανύσματά τους. Αυτό γίνεται μέχρι η πιθανότητες να φτάσει στο 1. Για το δοθέν κείμενο η έξοδος της συνάρτησης είναι η εξής:

```
Columns 1 through 8

  {'a'   }    {'b'     }    {'c'    }    {'d'    }    {'e'  }    {'f'     }    {'g'     }    {'h'    }
  {'0110'}    {'101111'}    {'10001'}    {'10110'}    {'110'}    {'100100'}    {'101110'}    {'10011'}

Columns 9 through 16

  {'i'   }    {'j'       }    {'k'      }    {'l'    }    {'m'    }    {'n'   }    {'o'   }    {'p'    }
  {'0101'}    {'111110011'}    {'11111000'}    {'01000'}    {'10000'}    {'0111'}    {'0000'}    {'11110'}

Columns 17 through 24

  {'q'       }    {'r'   }    {'s'   }    {'t'   }    {'u'     }    {'v'     }    {'w'     }    {'x'     }
  {'111110010'}    {'1010'}    {'1110'}    {'0001'}    {'010010'}    {'100101'}    {'01000110'}    {'01001111'}

Columns 25 through 27

  {'y'     }    {'z'     }    {' '  }
  {'111111'}    {'1111101'}    {'001'}
```

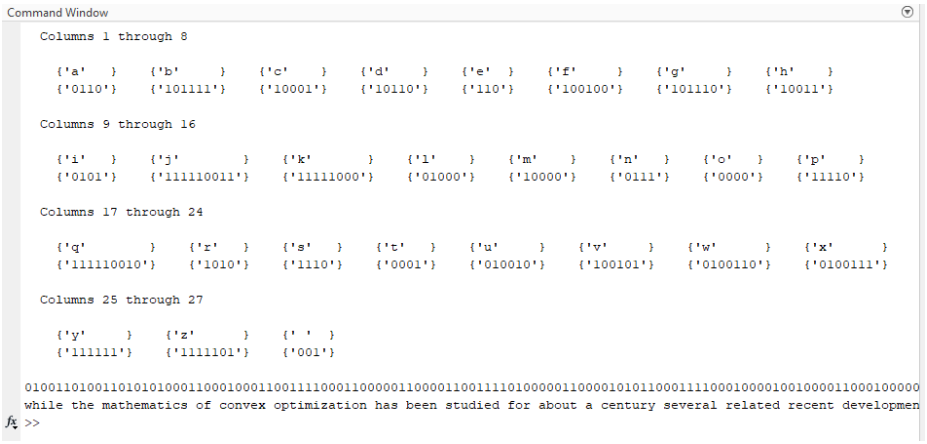
Εικ 1: Huffman Dictionary Output.

Huffman Encoding Η συνάρτηση huffmanEnc λαμβάνει ως είσοδο το αρχείο που θέλουμε να κωδικοποιηθεί και το dictionary που δημιουργήθηκε από την συνάρτηση huffmanDict. Όσο το αρχείο εισόδου δεν είναι κενό, συγκρίνει έναν προς έναν τους χαρακτήρες με την αντίστοιχη κωδικοποίησή τους και δημιουργεί ένα διάνυσμα εξόδου με την ακολουθία από bit που το αναπαριστούν. Τα αρχικά ψηφία φαίνοντα παρακάτω και μπορούν να επαληθευτούν από το dictionary.

```
Command Window
  Columns 1 through 8

    {'a'   }    {'b'     }    {'c'    }    {'d'    }    {'e'  }    {'f'     }    {'g'     }    {'h'    }
    {'0110'}    {'101111'}    {'10001'}    {'10110'}    {'110'}    {'100100'}    {'101110'}    {'10011'}

  Columns 9 through 16

    {'i'   }    {'j'       }    {'k'      }    {'l'    }    {'m'    }    {'n'   }    {'o'   }    {'p'    }
    {'0101'}    {'111110011'}    {'11111000'}    {'01000'}    {'10000'}    {'0111'}    {'0000'}    {'11110'}

  Columns 17 through 24

    {'q'       }    {'r'   }    {'s'   }    {'t'   }    {'u'     }    {'v'     }    {'w'     }    {'x'     }
    {'111110010'}    {'1010'}    {'1110'}    {'0001'}    {'010010'}    {'100101'}    {'01000110'}    {'01001111'}

  Columns 25 through 27

    {'y'     }    {'z'     }    {' '  }
    {'111111'}    {'1111101'}    {'001'}

0100110100110101010001100010001100111110001100000110000110011110100000110000101011000111100010000100100001100010000
```

Εικ 2: Huffman Encoding Output.

Huffman Decoding Η συνάρτηση huffmanDec λαμβάνει ως είσοδο το κωδικοποιημένο διάνυσμα όπως επίσης και το dictionary που έχει δημιοργηθεί. Διαβάζοντας έναν έναν τα bit εισόδου, δημιουργεί ένα προσωρινό λεξιλόγιο με σκοπό να ταυτοποιήσει την ακολουθία από bit που αντιστοιχεί στο κάθε γράμμα. Όταν το εντοπίσει, το γράμμα εισέρχεται στο vector εξόδου. Παρακάτω φαίνονται τα αποτελέσματα όλων των συναρτήσεων:



Εικ 3: Huffman Decoding Output.

1.2 Αποδοτικότητα κώδικα

Για να υπολογίσουμε την αποδοτικότητα του κώδικα, πρέπει πρώτα να υπολογιστούν η εντροπία του κώδικα όπως επίσης και το μέσο μήκος κώδικα. Ο τύπος της εντροπίας είναι:

$$\sum iPx(i) * log(Px(i)) \tag{1}$$

και ο τύπος του μέσου μήκους κώδικα είναι:

$$\sum iPx(i) * Length(i)) \tag{2}$$

Υπολογίζοντας αυτά μπορώ να βρώ την αποδοτικότητα του κώδικα αφού αποτελεί απλώς μια διαίρεση μεταξύ εντροπίας και μήκους. Τα αποτελέσματα φαίνονται παρακάτω:

```
while the mathematics of convex optimization has been s

entropy =

    4.1347


avgLenCode =

    4.1630


efficiency =

    0.9932

>>
```

Εικ 4: Huffman Efficiency

Όπως φαίνεται η αποδοτικότητα του κώδικα είναι στο 99% το οποίο είναι και ένα αποτέλεσμα που περιμέναμε αφού οι πιθανότητες που χρησιμοποιώ προέρχονται από το κείμενο που έχει κωδικοποιηθεί

1.3 Κωδικοποίηση με πιθανότητες συμβόλων αγγλικής γλώσσας

Χρησιμοποιώντας το αρχείο frequencies.txt για τις πιθανότητες των συμβόλων λαμβάνω το εξής αποτέλεσμα:

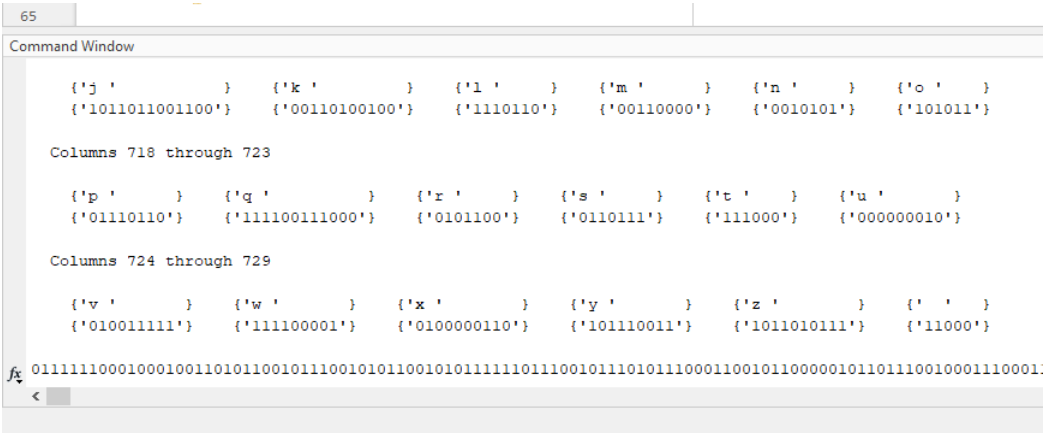
```
entropy =

    4.1679


avgLenCode =

    4.1875


efficiency =

    0.9953

>>
```

Εικ 5: Huffman Efficiency

Παρατηρώ ότι το μέσο μήκος κώδικα είναι λίγο μεγαλύτερο, το οποίο οφείλεται στο γεγονός ότι οι δοσμένες πιθανότητες είναι γενικευμένες για την αγγλική γλώσσα και δεν είναι προσαρμοσμένες πάνω στο δοθέν κείμενο.

1.4 Δεύτερη τάξη επέκταση πηγής A

Αρχικά δημιουργώ την δεύτερη τάξη επέκταση πηγής A, η οποία είναι όλοι οι συνδυασμοί γραμμάτων μεταξύ τους και υπολογίζω τις πιθανότητες κάθε συνόλου 2 χαρακτήρων πολλαπλασιάζοντας τις πιθανότητες του καθενός. Στο τέλος καταλήγω με 729 συνδυασμούς γραμμάτων (δηλαδή $26\hat{2}$). Η τελική κωδικοποιηση φαίνεται παρακάτω μαζί με τα τελευταία ζεύγη του λεξιλογίου:



Εικ 6: Huffman Second Order

Παρατηρώ ότι παρόλο που το μέσο μήκος κώδικα έχει διπλασιαστεί, το οποίο είναι και ένα αποτέλεσμα που περιμέναμε λόγω της αύξησης των χαρακτήρων από 27 σε 729, η απόδοση της κωδικοποίησης είναι μεγαλύτερη και η συμπίεση του κειμένου είναι μεγαλύτερη. Επίσης η εντροπία μια διακριτής πηγής με την αντίστοιχη n τάξης επέκτασής της ακολουθεί τον τύπο:

$$H(\Phi n) = nH(\Phi) \tag{3}$$

Το οποίο επαληθεύεται με τα δεδομένα μας, αφού η εντροπία της δεύτερης τάξης επέκταση της πηγής A είναι διπλάσια.

1.5 Κωδικοποίηση εικόνας και χρήση δυαδικού συμμετρικού καναλιού

Αρχικά φορτώνω σε ένα struct το αρχείο cameraman.mat και υπολογίζω τις πιθανότητες των τιμών 0-255 που αποτελούν την κλίμακα του γκρι. Έπειτα δημιουργείται το dictionary για την εικόνα και κωδικοποιείται η πηγή. Για να βρούμε την

πιθανότητα p κάποιου bit να έχει μετατραπεί από 0 σε 1 ή από 1 σε 0 αρκεί να βρούμε την απόσταση hamming των δύο διανυσμάτων. Έπειτα, γνωρίζοντας την πιθανότητα μπορούμε να υπολογίσουμε την δυαδική εντροπία και τέλος η χωρητικότητα του καναλιού είναι η

$$C = 1 - H(p) \tag{4}$$

Τα αποτελέσματά μου φαίνονται παρακάτω:

```
95    % find hamming distance of 2 strings
96    p = pdist2(encImage, noisyImg, 'hamming')
97    %calculate binary entropy
98    H = -p*log2(p) - (1-p)*log2(1-p)
99    %Capacity of channel
100   C = 1 - H
101
```

```
Command Window
  p =

      0.1799


  H =

      0.6798


  C =

      0.3202

fx >>
```

Εικ 7: Channel Capacity

2 Ερώτημα 2 - Κωδικοποίηση PCM

Η κωδικοποίηση PCM χρησιμοποιείται για την κωδικοποίηση αναλογικών σημάτων σε ψηφιακά. Σκοπός αυτής της άσκησης είναι η δημιουργία ενός μη ομοιόμορφου βαθμωτού κβαντιστή που θα υλοποιηθεί χρησιμοποιώντας τον αλγόριθμο Lloyd-Max.

2.1 Ερώτηση 1 - Κωδικοποίηση πηγής Α

Η πηγή Α είναι μια πηγή που ακολουθεί την κανονική κατανομή για Μ = 10000 χρησιμοποιώντας την συνάρτηση:

$$t = randn(M, 1) \tag{5}$$

Παρακάτω φαίνονται οι έξοδοι για Ν = 2, 3 και 4 bits αντίστοιχα. Τα διανύσματα παραμόρφωσης είναι ενδεικτικά και δε φαίνονται ολόκληρα στις εικόνες.

```
     1
     2
     3
     2
     2
     4

centers =

   -0.4038   -0.1307    0.1078    0.3921

D =

  Columns 1 through 12

    1.0000    0.0976    0.0861    0.0796    0.0760    0.0740    0.0731    0.0726    0.0724    0.0722    0.0722    0.0721

  Columns 13 through 24

    0.0721    0.0721    0.0721    0.0721    0.0721    0.0721    0.0721    0.0721    0.0721    0.0721    0.0721    0.0721

  Column 25

    0.0721

>>
```

Εικ 8: Source A with 2 bit zones

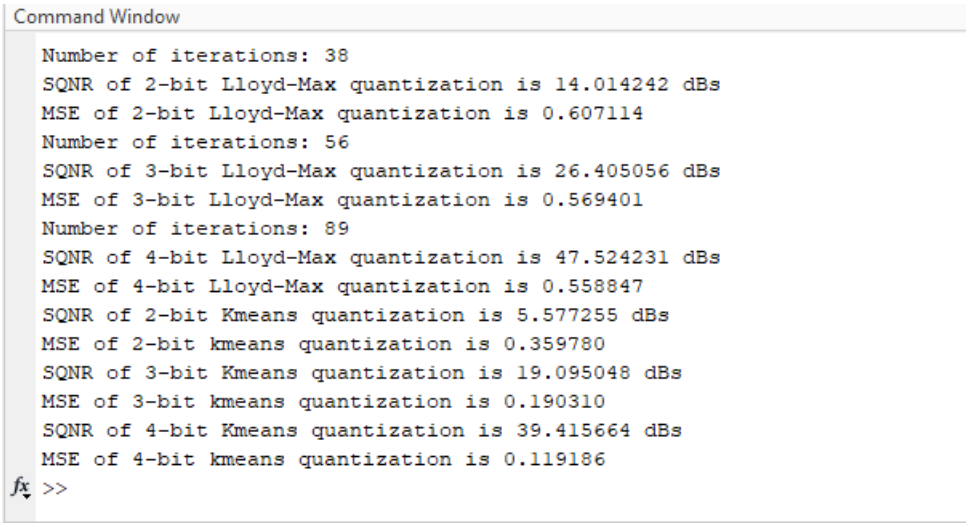
```
Command Window
     6
     8
     7
     1

centers =

   -0.5845   -0.3570   -0.1962   -0.0596    0.0744    0.2106    0.3687    0.5939

D =

  Columns 1 through 12

    1.0000    0.0577    0.0550    0.0530    0.0512    0.0498    0.0485    0.0473    0.0462    0.0453    0.0445    0.0438

  Columns 13 through 24

    0.0434    0.0429    0.0425    0.0422    0.0419    0.0416    0.0414    0.0411    0.0410    0.0408    0.0407    0.0406

  Columns 25 through 36

    0.0406    0.0405    0.0405    0.0404    0.0404    0.0404    0.0404    0.0403    0.0403    0.0403    0.0403    0.0403

  Columns 37 through 48

    0.0403    0.0403    0.0403    0.0403    0.0403    0.0403    0.0402    0.0402    0.0402    0.0402    0.0402    0.0402
```

Εικ 9: Source A with 3 bit zones

```
Command Window
    10
     8
    14
     5
    14
     4

centers =

  Columns 1 through 12

   -0.7367   -0.5789   -0.4534   -0.3560   -0.2775   -0.2044   -0.1355   -0.0702   -0.0066    0.0583    0.1256    0.1973

  Columns 13 through 16

    0.2748    0.3640    0.4879    0.6812

D =

  Columns 1 through 12

    1.0000    0.0308    0.0302    0.0297    0.0292    0.0288    0.0283    0.0278    0.0273    0.0268    0.0263    0.0259

  Columns 13 through 24

    0.0255    0.0251    0.0248    0.0246    0.0244    0.0242    0.0240    0.0238    0.0236    0.0234    0.0233    0.0232
```

Εικ 10: Source A with 4 bit zones

Υπολογίζοντας το SQNR για τον βαθμωτό και τον διανυσματικό κβαντιστή λαμβάνουμε τα εξής αποτελέσματα:

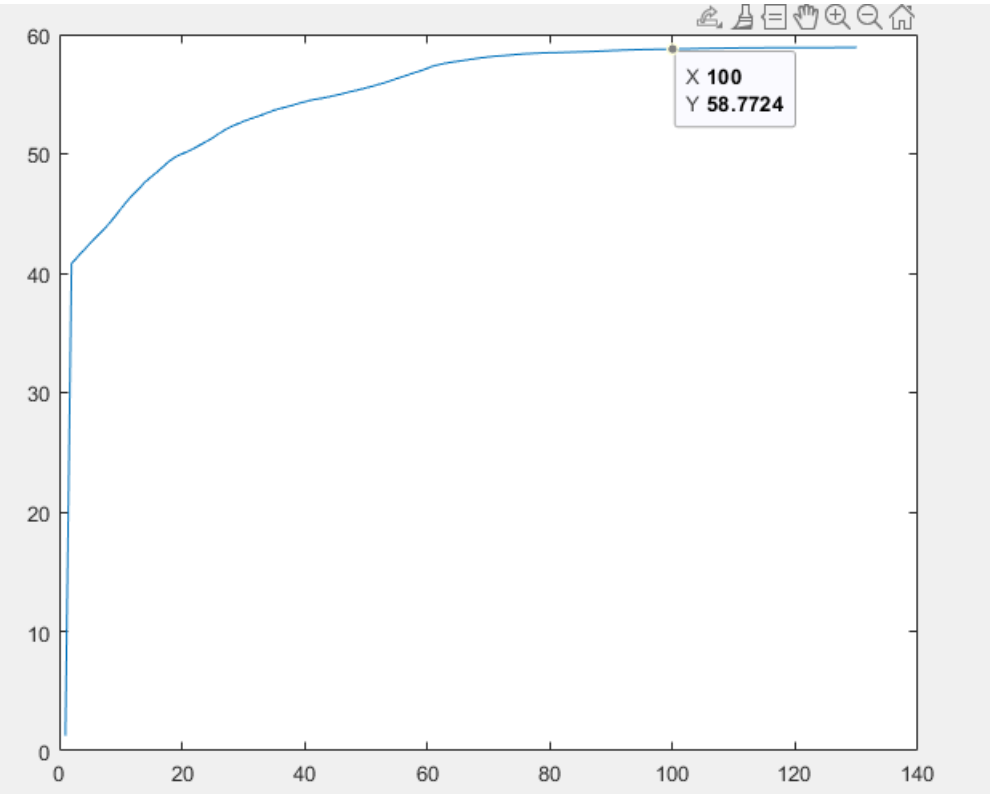


Εικ 11: SQNR Source A

Όπως βλέπουμε το SQNR αυξάνεται όσο αυξάνονται τα επίπεδα κβάντισης του κάθε αλγορίθμου. Επίσης βλέποντας το MSE ανάμεσα στο αρχικό και κβαντισμένο διάνυσμα βλέπουμε ότι ο θόρυβος που δημιουργείται λόγω της κβάντισης μειώνεται με όσο περισσότερα bit χρησιμοποιούμε για τα επίπεδα κβάντισης.

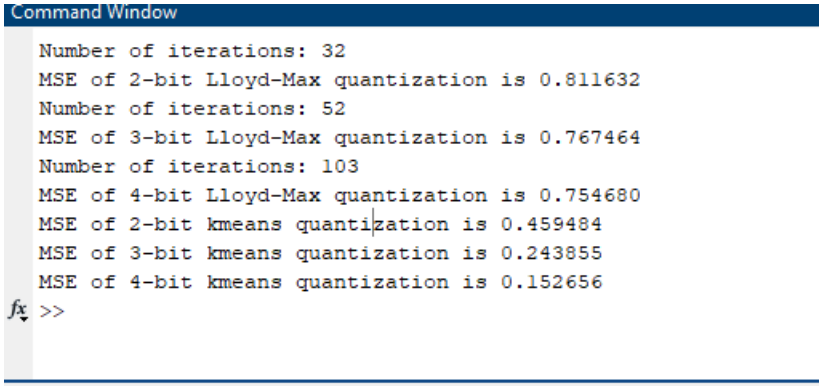
2.2 Ερώτηση 2 - Κωδικοποίηση πηγής B

Χρησιμοποιώντας την πηγή B, το διάγραμμα που λαμβάνουμε σχεδιάζοντας το SQNR σε σχέση με τον αριθμό των επαναλήψεων είναι το εξής:



Εικ 12: SQNR Source B Diagram

Επίσης μετρώντας το MSE των δύο αλγορίθμων βλέπουμε ότι το λάθος που παράγεται από τον διανυσματικό χβαντιστή είναι αρκετά μικρότερο από τον βαθμωτό.



Εικ 13: MSE of the 2 quantizers