ACTIVE USER DETECTION AND LOW COMPLEXITY MULTI USER DETECTION FOR UNSOURCED MULTIPLE ACCESS

A Thesis

by

NICHOLAS JEON

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2021

Major Subject: Electrical and Computer Engineering

ABSTRACT

We consider two problems related to uncoordinated multiple access. The first is the design of schemes that can identify the set of active users. The second is the design of low complexity multiuser detection schemes. We consider a multiple access scheme where each user encodes its information by an error correcting code and spreads the coded bits using a spreading sequence that is chosen from a master set of spreading sequences based on part of each user's message. On the receiver side, the set of active spreading sequences need to identify first and then the bits of the user have to be detected and decoded.

We consider four strategies for evaluating the set of active spreading sequences. The first scheme is a correlation-based energy detector. The second scheme is based on machine learning, which uses the histogram of the output of a matched-filter(MF) as input to a neural network(NN) model. The third scheme is based on using a hypothesis test on the outputs of the matched filter. The fourth scheme is a 2-bit combined energy detector, which is the same way for the original energy detector but combining two bits to consider more about the variable case that can happen in synchronizing spreading sequences.

In the second part of the thesis, assuming the set of active sequences is known, an MMSE estimator is implemented to perform log-likelihood ratios(LLRs) for the active sequences. But inverting the whole active sequences matrix in MMSE has large time complexity. We propose using a clustering method to reduce matrix size. After this active sequences are passed to a list decoder of polar code proceeding iteratively by subtracting the interference due to the successfully decoded sequence from the received signal and repeat the MMSE estimator process on the residual received signal. The cluster size and a decoding schedule are optimized using Monte Carlo simulations.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Krishna Narayanan of the Department of Electrical and Computer Engineering.

The data analyzed for Chapter 2.2 and 3.1 was provided by Professor Krishna Narayanan. The analyses depicted in Chapter 2.2 and 3.1 were conducted in part by Asit Kumar Pradhan of the Department of Electrical and Computer Engineering and were published in 2019 in an article listed in the arXiv.org.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

There are no outside funding contributions to acknowledge related to the research and compilation of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Unsourced multiple access provides a model for a novel communication paradigm attuned to machine-type communication. In this new scheme, there are K_{tot} users in a wireless communication system whithin which a smaller group of K_a users are active at any time. These K_a users each have a length B-bit message that is to be transmitted using a multiple access channel. In [1], the authors capture scenarios where a large number of devices transmit short packets of information in an intermittent manner to a central unit. Ever since the introduction of unsourced multiple access communication(MAC), there have been numerous attempt in designing algorithm schemes to operate close to the finite-blocklength boundary while maintaining computational complexity like [1–3].

In this thesis, we compare four techniques for determining the set of active sequences. First using the correlation-based energy detector to determine which sequences are most likely to be active. Similarly combining the next bit for users to shorten the length of B to make a 2-bit combined energy detector to figure out the active sequences. The third method is we use the histogram from the matched filter(MF) to stack the data into the input to a neural network. In this neural network model, the structure is a fully-connected layer with two inputs. Another input is the value of the energy detector. The last method is that from MF we concentrate on the distribution of each user to compare likelihood function for active and inactive sequence. Using all the schemes above to detect active sequence, the MMSE estimator is implemented to recover the original signal. But inverting the whole size of the MMSE matrix is complex. Thereby we use the clustering method to shrink down the size of the matrix to get lower time complexity.

The rest of this thesis is organized as follows. In Chatper 2, we introduce the system model for unsourced multiple access that we will consider in the rest of this thesis. In Chapter 3, we introduce two algorithms for detecting the set of active users. In Chatper 4, we propose a clustering based algorithm for reducing the complexity of minimum mean-squared error estimation of the transmitted bits.

From the received signal y, identify the active users with 4 schemes, which is in chapter 3. After knowing the expected active user's list, we estimate the transmitted bits using the MMSE equation and try to reduce the complexity in chapter 4. In the decoder block, we implemented polar code introduced in [2] to find out which user can be successfully decoded and removed from the received signal y.

2. SYSTEM MODEL

In the random access model, let K_{tot} be the total number of users in the network and let K_a denote the subset of active users(each active user wishes to communicate B bits of a message to an n uses of the base station). This data must be transmitted using an unsourced uplink transmission scheme. That is, the data of each user are not correlated and active users must act independently of one another. Let s_i denote as indicator random variable which is 1 if the user i is active, and 0 otherwise. The received signal, $\vec{y}$, at the base station is given by

$$\vec{y} = \sum_{i=1}^{K_{tot}} s_i \vec{x_i}(w_i) + \vec{z} \tag{1}$$

where $w_i \in \{0,1\}^B$ equivalent to the B bit message that user i try to transfer to the base station, $\vec{x_i}$ denotes the N-dimensional vector transmitted by user i, $\vec{z} \sim N(0, \sigma^2 I)$ is additive white Gaussian noise (AWGN), and $\sum_{i=1}^{K_{tot}} s_i = K_a$. It is assumed that when $s_i = 1$, user i selects its message W_i uniformly from the set $[M] \triangleq [1 : M](M - 2^B)$. Each active users are independent to one another, which mean that the messages chosen by the active users has no correlation to one and other for K_a. The signal sent by a device is power constrained, $\| \vec{x_i} \|_2^2 \leq nP$ for $i \in K_a$, a case similar to [1]. The energy-per-bit of system is defined as $\frac{E_b}{N_o} = \frac{nP}{2B}$. From the received signal, the decoder produces an estimate $\hat{\theta}(\vec{y})$ for the transmitted binary vectors θ with size at most K_a. From original equation of [1], the per-user error probability is given by

$$P_e = \max_{\sum s_i = K_a} \frac{1}{K_a} \sum_{i=1}^{K_{tot}} s_i Pr(w_i \notin \hat{\theta}(\vec{y})) \tag{2}$$

for fixed values of n, B, K_a, ε. Our goal is to design a low complexity scheme that achieves $P_e \leq \varepsilon$, where ε is our goal error probability at required $\frac{E_b}{N_o}$.

2.1 Encoder

For a specific user, the user wishes to transmit B bits and $M_s = 2^{B_s}$. We denote the message corresponding to user i by w_s that is, $w = w_s$. The spreading sequence used by the active users are determined by following. Let A = $[\vec{a_1}, \vec{a_2}, ..., \vec{a_{M_s}}] \in \mathbb{R}^{n_s \times M_s}$ denote all the possible spreading sequence. The elements of A are generated with zero mean Gaussian random variables with unit variance and chosen by drawing independent. We assumed A also satisfy the power constraint. Every active user inside the set of K_a is assigned one column inside of matrix A as the spreading sequence. That is to say, each active user use function f : $\{0, 1\}^{B_s} \rightarrow \{\vec{a_j} : j \in [1 : M_s]\}$. We denote the spreading sequence chosen by the active user i as $\vec{a_{j_i}}$. At the end, each codeword is encrypted by a spreading sequence to generate the transmitted signal. The signal transmitted by the active user i is given by

$$\vec{x_i} = v_i \otimes \vec{a_{j_i}} \tag{3}$$

where v_i is the codeword and $\otimes$ denotes tensor product symbol. We assumed that $\vec{x_i}$ satisfies the power constraint $\| \vec{x_i} \|_2^2 \leq P$ for any messages.

2.2 Decoder

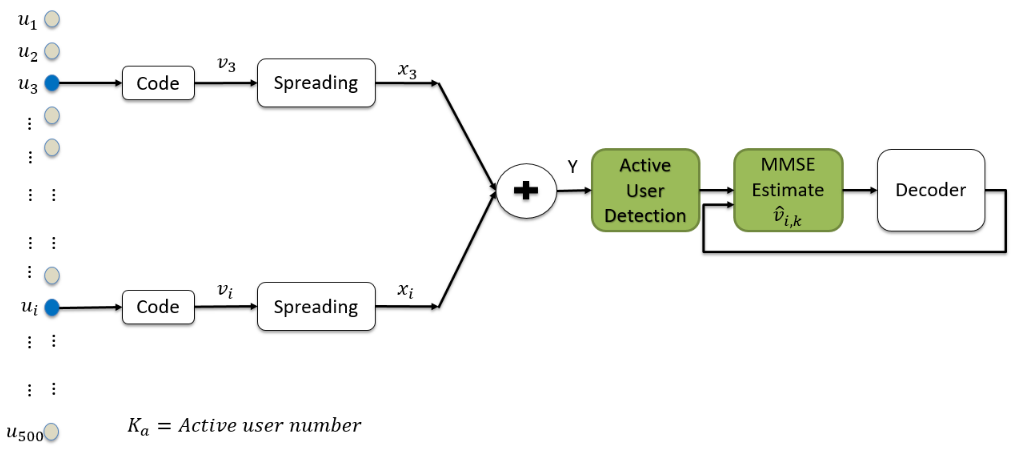


Figure 1: Block diagram for the system

Figure 1 shows the outline for the system and this thesis is going to focus on how to identify the active users and estimate what was messages were transmitted. The decoding process has two distinct stages. During the first stage, we have to identify the set of spreading sequences employed by the active users. In the second stage, a minimum mean squared error (MMSE) estimator is employed to produce estimates of the coded symbols. In this step, the decoder employs a reduced complexity MMSE estimator which clusters the most correlated sequences to reduce the dimension of the matrix need to be inverted. These estimated sequences then passed on to the decoder of polar code like in [2–4]. The signal codeword corresponding to the successfully decoded users are removed from the received signal. Then the remaining signal redirected to the sequence detector.

3. ACTIVITY DETECTION

In this chpater, we design two algorithms to identify active sequences. They are - 1) Neural Network model, and 2) Hypothesis testing. We compare the performance of these algorithms with the matched filter and bit-combined matched filter from earlier papers in [2,3]. To compare each of these algorithms, we compare the probability of false alarm and miss detection of these schemes.

3.1 Energy Detector and 2 bit combined Energy Detector

The purpose of the energy detector scheme is to detect which sequence is active based on statistics that incorporates energy. In the encoder part where $\vec{x_i} = v_i \otimes \vec{a_{j_i}}$, and try to decode $\vec{x_i}$ but v_i is unknown. One way to approach this would be the matched filter(MF) in [2]. MF is a basic tool for extracting known transmitted bits from a signal that has been contaminated by noise. The output of the MF can be obtained by correlating $\vec{y}$ with all the given spreading sequences.

$$r = A^T \times \vec{y} \tag{4}$$

For example, if we look at the 1_{st} row and 1_{st} column of the matrix r, which is assigned with a first spreading sequence is given by

$$\begin{aligned} r_{1,1} &= a_1 \vec{y_1} \\ &= a_{1,1}^2 v_{1,1} + a_{1,1}(a_{2,1}v_{2,1} + a_{3,1}u_{3,1} \ldots a_{K_{tot},1}v_{K_{tot},1}) \\ &\qquad\qquad \ldots \\ &\qquad\qquad + \\ &\quad a_{1,B}^2 v_{1,1} + a_{1,B}(a_{2,B}v_{2,1} + a_{3,B}u_{3,1} \ldots a_{K_{tot},B}v_{K_{tot},1}) \\ &= \vec{v}_{1,1} + \sum_{k=1}^{B} a_{1,k}(a_{2,k}v_{2,1} + a_{3,k}u_{3,1} \ldots a_{k_{tot},B}u_{k_{tot},1}) \end{aligned}$$

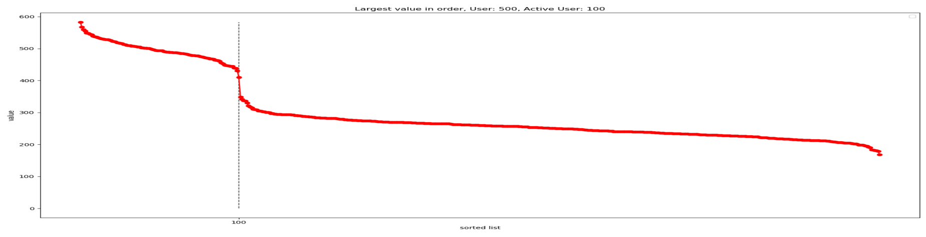


Figure 2: An example of sorted user base on energy detector.

where, $\vec{y_1}$ is the first column of matrix y and $\vec{v}$ is assigned with -1,1 if active users or 0 if inactive. From this MF equation, the energy is given by

$$E_{i,j} = \sum_{j=1}^{B} r_{i,j}^2 \tag{5}$$

where i and j represent the i_{th} user and location of j_{th} bit. From equation (5), it is possible to make the certain gap between active users and inactive users because $\vec{v}_{i,j}$ can be extracted from the equation. As seen from an example in Figure 2, in this figure $K_a = 100$. we can visualize there is a certain gap in 100_{th} user. Most active users are at the left side of the gap with few inactive users. Based on these statistics, The energy detector gives us the first K sequences in the sorted list, where K = $K_a + K_\delta$ for fixed integer K_δ. With the higher value of the energy detector, it will have more chances to be an active sequence. 2 bit combined energy detector is similar to the previous scheme. As shown in Figure 3, reshaping the received signal to the length of $\frac{B}{2}$ we can consider four cases. $[a_i a_i, a_i - a_i, -a_i a_i, -a_i - a_i]$. For each case correlate with $\vec{y}$ and repeat the step as done in the original energy detector.

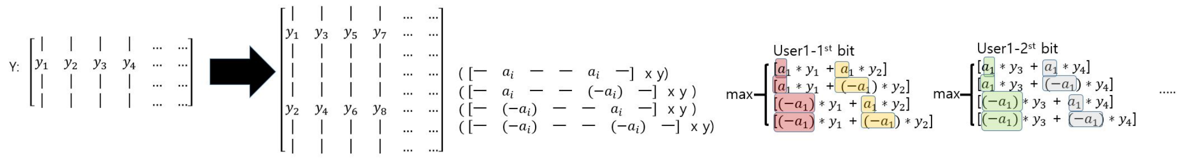

Y:
y_1 y_2 y_3 y_4
y_1 y_3 y_5 y_7
y_2 y_4 y_6 y_8
([− a_i − − a_i −] x y)
([− a_i − − $(-a_i)$ −] x y)
([− $(-a_i)$ − − a_i −] x y)
([− $(-a_i)$ − − $(-a_i)$ −] x y)
max
User1-1st bit
$[a_1 * y_1 + a_1 * y_2]$
$[a_1 * y_1 + (-a_1) * y_2]$
$[(-a_1) * y_1 + a_1 * y_2]$
$[(-a_1) * y_1 + (-a_1) * y_2]$
max
User1-2st bit
$[a_1 * y_3 + a_1 * y_4]$
$[a_1 * y_3 + (-a_1) * y_4]$
$[(-a_1) * y_3 + a_1 * y_4]$
$[(-a_1) * y_3 + (-a_1) * y_4]$
.....

Figure 3: 2bit combined MF for the u_1

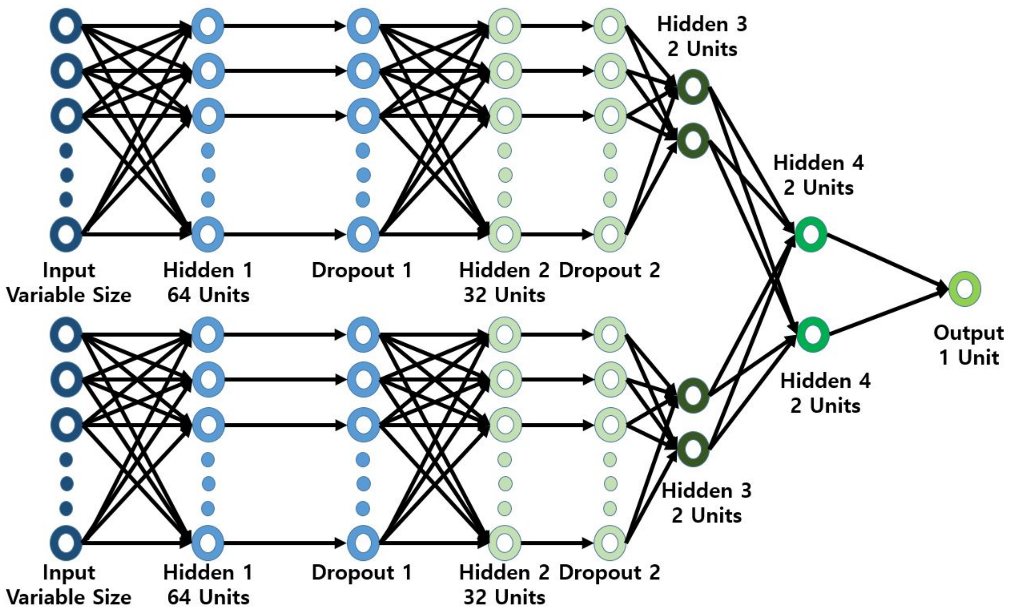


Figure 4: Neural Network structure

3.2 Deep Learning based active user detector

Deep learning has two separate operating phases. A training part followed by a testing phase. The goal of the training is to optimize and prepare the NN for the test samples. After training, it is in the test phase that the NN's functionality as a detector is realized. In both phases, the energy detector and histogram of MF are fed as an input to the NN. Because the NN model has multiple inputs, the structure we used is parallelism as in Figure 4. As mentioned in [5], the NN has multiple layers with parameters, called weights and biases, these parameters are optimized at the training phase. The layers used in Figure 4 is called the fully-connected layer and activation layer. The activation layer introduces non-linearity and it has no trainable parameters. Mostly activation function for the hidden layers is the Rectified Linear Unit(ReLU). ReLU activation is sequentially repeated before getting to the output. For the output layer of a classifier, sigmoid activation is selected because at the end we only have one output to label it. In other words, the number of output nodes is going to determine the activation for the output layer. In the training phase, the dataset is sent to NN multiple times. Epoch is used to indicate how many times that the dataset was sent through NN. Let say the number of the epoch is N_{epoch} and let N_{batch} denote that to avoid overflow in the training phase, the dataset is divided into several smaller subsets. N_{batch} called the batch-size. The number of epochs, the batch-size, and the learning rate α are called the hyper-parameters. In the test phase, the performance is determined by the training phase convergence, which depends on these hyper-parameters. In our NN, a loss function, called binary cross entropy is used and is given by

$$\mathcal{L}(\tilde{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} \theta \log_2(\tilde{\theta}) + (1-\theta) log_2(1-\tilde{\theta}) \tag{6}$$

where θ is the label and the NN's output layer activation is the sigmoid function. Let say an input d from the previous layer, the sigmoid function is given by

$$\tilde{\theta} = \frac{1}{1+e^{-d_m}}, \forall m \tag{7}$$

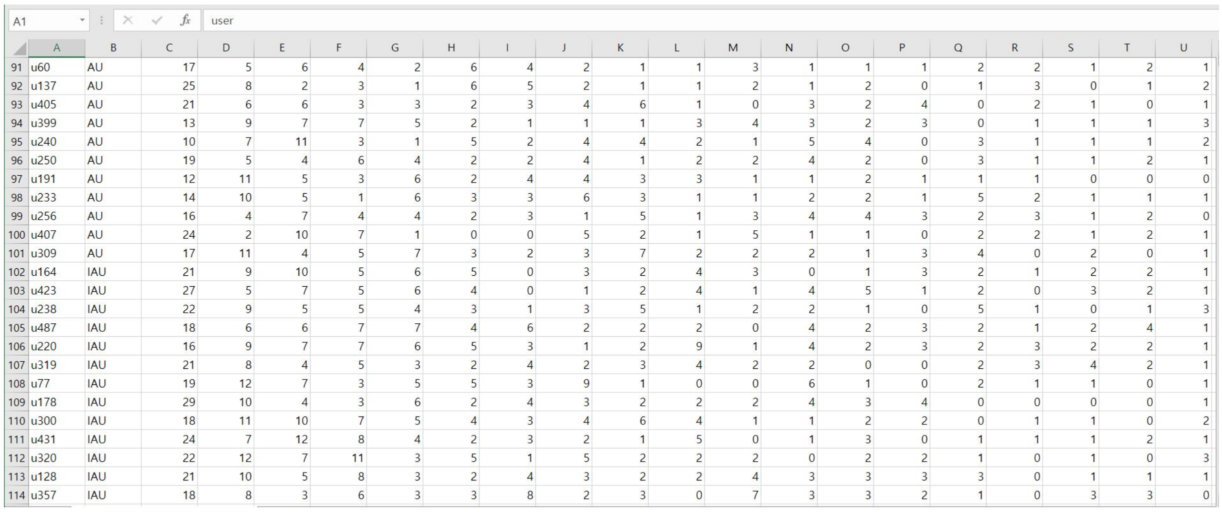
and the hard decision is perform as round integer, which is given by

$$\hat{\theta} = \begin{cases} 0 \text{ if } \tilde{\theta} \leq 0.5, \\ 1 \text{ otherwise.} \end{cases} \tag{8}$$

3.2.1 Training data and Test data

In a dataset, usually divided into a training set, a validation set, and a test set in each iteration. A training set is implemented to build up a model. In a neural network, we try to create a model to predict the test data. In order to do that we use the training data to fit the model and testing data to test it. The models generated are to predict the results unknown which is a test set. The proportion between train data and test data is up to the designer.

In this thesis, we used a multi-input system for the training sample. The first input is the histogram for the square value of output of the MF and the other one is the energy detector of each user. The histogram is an approximate representation of the distribution of numerical data. To



A1 user

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
91	u60	AU	17	5	6	4	2	6	4	2	1	1	3	1	1	1	2	2	1	2	1
92	u137	AU	25	8	2	3	1	6	5	2	1	1	2	1	2	0	1	3	0	1	2
93	u405	AU	21	6	6	3	3	2	3	4	6	1	0	3	2	4	0	2	1	0	1
94	u399	AU	13	9	7	7	5	2	1	1	1	3	4	3	2	3	0	1	1	1	3
95	u240	AU	10	7	11	3	1	5	2	4	4	2	1	5	4	0	3	1	1	1	2
96	u250	AU	19	5	4	6	4	2	2	4	1	2	2	4	2	0	3	1	1	2	1
97	u191	AU	12	11	5	3	6	2	4	4	3	3	1	1	2	1	1	1	0	0	0
98	u233	AU	14	10	5	1	6	3	3	6	3	1	1	2	2	1	5	2	1	1	1
99	u256	AU	16	4	7	4	4	2	3	1	5	1	3	4	4	3	2	3	1	2	0
100	u407	AU	24	2	10	7	1	0	0	5	2	1	5	1	1	0	2	2	1	2	1
101	u309	AU	17	11	4	5	7	3	2	3	7	2	2	2	1	3	4	0	2	0	1
102	u164	IAU	21	9	10	5	6	5	0	3	2	4	3	0	1	3	2	1	2	2	1
103	u423	IAU	27	5	7	5	6	4	0	1	2	4	1	4	5	1	2	0	3	2	1
104	u238	IAU	22	9	5	5	4	3	1	3	5	1	2	2	1	0	5	1	0	1	3
105	u487	IAU	18	6	6	7	7	4	6	2	2	2	0	4	2	3	2	1	2	4	1
106	u220	IAU	16	9	7	7	6	5	3	1	2	9	1	4	2	3	2	3	2	2	1
107	u319	IAU	21	8	4	5	3	2	4	2	3	4	2	2	0	0	2	3	4	2	1
108	u77	IAU	19	12	7	3	5	5	3	9	1	0	0	6	1	0	2	1	1	0	1
109	u178	IAU	29	10	4	3	6	2	4	3	2	2	2	4	3	4	0	0	0	0	1
110	u300	IAU	18	11	10	7	5	4	3	4	6	4	1	1	2	2	0	1	1	0	2
111	u431	IAU	24	7	12	8	4	2	3	2	1	5	0	1	3	0	1	1	1	2	1
112	u320	IAU	22	12	7	11	3	5	1	5	2	2	2	0	2	2	1	0	1	0	3
113	u128	IAU	21	10	5	8	3	2	4	3	2	2	4	3	3	3	3	0	1	1	1
114	u357	IAU	18	8	3	6	3	3	8	2	3	0	7	3	3	2	1	0	3	3	0

Figure 5: Histogram with bin = 0.1

construct a histogram, the first step is to bin the range of values and then count how many values fall into each interval. In this case, the square value of output of the MF with B bits for each user is observed data. Like above in Figure 5, each row represents a specific user histogram with the corresponding label and value. The second column will be used to classify the user's status and the rest of the columns will be trained in the model. In this process collecting large training data is one of the tasks. For each iteration, accumulate the fixed number of the histogram and feed these data into a neural network model. At the next iteration, reload the previous model and retrain with the current fixed number of histograms and repeat this step until the training data is sufficient. But every iteration we have to train the model and reload it, which is an inefficient way to approach it. Instead of reloading the previous model, at each iteration, save the fixed number of histograms in csv file, and at the next iteration reload the last csv file and stack with the new histogram. Approaching this way, we can reduce the processing time and train the model only once. For collecting another input, which is the value of the energy detector for each user is implemented in the same way as a histogram and combined with all the input for the model.

Accumulating the training data with different ratios between classes is another variable that has to be considered. Like Figure 6, there are several ways to collect the data. The best way to collect data is to set the equal ratio with each class for this particular model. In this paper, we collect all

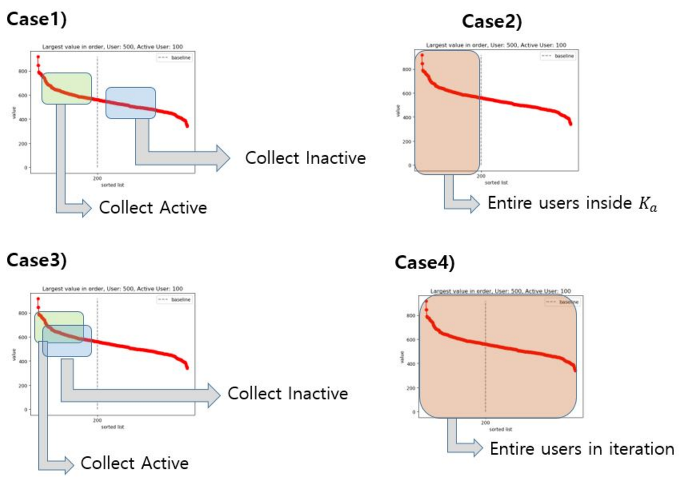

Case1)
Case2)
Case3)
Case4)
Largest value in order, User: 500, Active User: 100
baseline
value
sorted list
Collect Inactive
Collect Active
Entire users inside K_a
Entire users in iteration

Figure 6: Collecting training data with different ratio

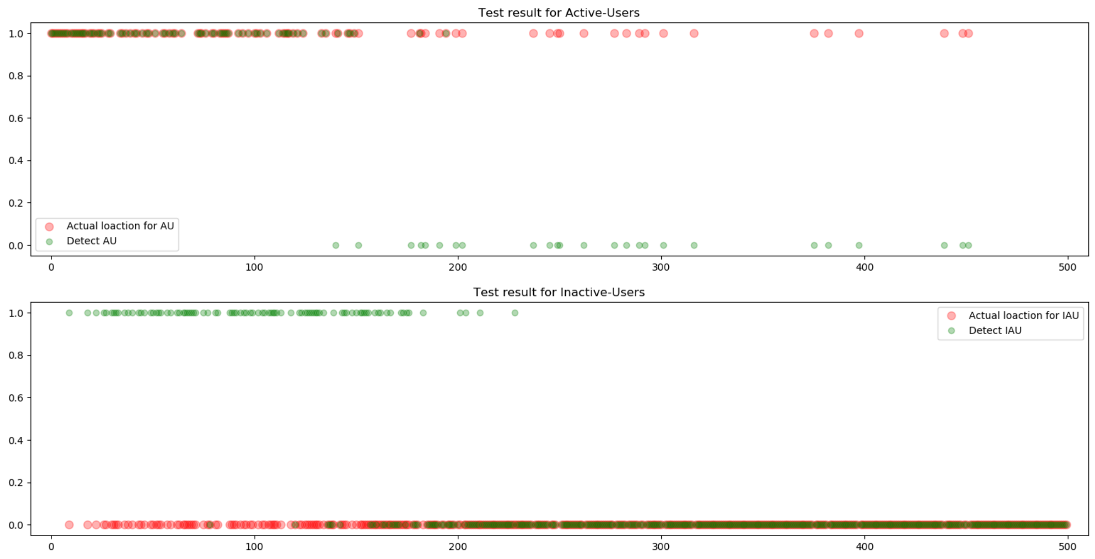


Figure 7: example of miss detection and false alarm

the active users and randomly pick the inactive users with the same ratio. Figure 7 is an example of the same ratio between active users and inactive users. It shows the miss detection for active sequences and false alarms for inactive sequences. After the K_a, it is difficult for action sequences to be detected. On the contrary, it is hard to identify inactive sequences before the K_a.

3.3 Hypothesis Testing

In this approach, we compute the likelihood of the square of the correlation of the observed signal with each spreading sequence under two hypotheses - the user is active and the user is inactive. Depending on whichever likelihood is higher, we decide the user is active or not. From the square value for the output of the MF, the formula for PDF of normal distribution is given by

$$f_X(x_i : u, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-u)^2}{2\sigma^2}} \tag{9}$$

where u and σ^2 are the parameters mean and variance of the normal distribution and x_i is observed value. In order to stress the fact that the probability density depends on the two parameters, we write the joint probability density of the sample ζ is

$$f(\zeta; \theta) = \prod_{i=1}^{n} f_X(x_i : u, \sigma^2) \tag{10}$$

Because the joint density of independent variables is same as the product of the marginal densities, the likelihood function is

$$L(\theta; \zeta) = f(\zeta; \theta) = \prod_{i=1}^{n} f_X(x_i : u, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-u)^2}{2\sigma^2}} \tag{11}$$

The log-likelihood function is

$$\begin{aligned} l(\theta; \zeta) &= \ln\left[L(\theta; \zeta)\right] \\ &= \ln\left[\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-u)^2}{2\sigma^2}}\right] \\ &= \sum_{i=1}^{n} \ln\left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-u)^2}{2\sigma^2}}\right] \end{aligned} \tag{12}$$

3.3.1 Inactive user

To compute the PDF of the square value of the output of the MF, we have to calculate the value u and σ^2 for observed value. The observed value R is given by

$$\begin{aligned} R_{i,k} &= (a_i^T y_k)^2 \\ &= (a_i^T a_i u_{i,k} + \sum_{j \neq i}^{K_t ot} a_i^T a_j u_{j,k} + a_i^T n_{i,k})^2 \\ &= (c_{i,k} + w_{i,k})^2 \end{aligned} \tag{13}$$

let assume that l_1 is length of the spreading sequence, $u_{i,k}$ is user message, $n_{i,k}$ is noise with i_{th} user and k_{th} location bit. In the equation, $c_{i,k}$represent the first term and $w_{i,k}$ represent second and third term.

$$E[w_{i,k}] = 0, E[w_{i,k}^2] = \frac{K_a}{l_1} + \sigma^2 \tag{14}$$

From the above equation, we can see that $w_{i,k}$ has a normal distribution with zero mean, $\frac{K_a}{l_1} + \sigma^2$ variance and $c_{i,k}$ is zero. In order to find the PDF of a random variable, it starts with the cumulative distribution function(CDF), which is integral of the PDF.

$$\begin{aligned} F_R(x) &= P(R \leq x) = P(W^2 \leq x) \\ &= P(-\sqrt{x} \leq W \leq \sqrt{x}) \\ &= F_W(\sqrt{x}) - F_W(-\sqrt{x}), now \quad take \quad differential \\ f_R(x) &= f_w(\sqrt{x})\frac{d}{dx}(\sqrt{x}) - f_w(-\sqrt{x})\frac{d}{dx}(-\sqrt{x}) \\ &= \frac{1}{2\sqrt{x}}\left[f_W(\sqrt{x}) + f_W(-\sqrt{x})\right] \\ &= \frac{1}{2\sqrt{x}}\frac{1}{\sqrt{2\pi(\frac{K_a}{l_1} + \sigma^2)}}e^{-\frac{x}{2(\frac{K_a}{l_1}+\sigma^2)}} \end{aligned} \tag{15}$$

3.3.2 Active user

For active user PDF, $R_{i,k}$ is same as inactive user case in equation (13). But variance value for $w_{i,k}$ is different from previous. Because from (13) we extract one user out from the equation, which make

$$c_{i,k} = 1, E[w_{i,k}] = 0\, E[w_{i,k}^2] = \frac{K_a - 1}{l_1} + \sigma^2 \tag{16}$$

Likewise, to compute the PDF of the square value of the output of the MF, start with CDF.

$$\begin{aligned}
Q &= U + W \\
f_Q(x) &= f_U(x) * f_W(x) \\
&= \frac{1}{2}N(-1, \frac{K_a - 1}{l_1} + \sigma^2) + \frac{1}{2}N(1, \frac{K_a - 1}{l_1} + \sigma^2) \\
R &= Q^2 \\
F_R(x) &= P(R \leq x) = P(Q^2 \leq x) \\
&= P(-\sqrt{x} \leq Q \leq \sqrt{x}) \\
&= F_Q(\sqrt{x}) - F_Q(-\sqrt{x}), \quad now \quad take \quad differential \\
f_R(x) &= f_Q(\sqrt{x})\frac{d}{dx}(\sqrt{x}) - f_Q(-\sqrt{x})\frac{d}{dx}(-\sqrt{x}) \\
&= \frac{1}{4\sqrt{x}} \frac{1}{\sqrt{2\pi(\frac{K_a-1}{l_1} + \sigma^2)}} \left[e^{\frac{(\sqrt{x}-1)^2}{2(\frac{K_a-1}{l_1}+\sigma^2)}} + e^{\frac{(-\sqrt{x}-1)^2}{2(\frac{K_a-1}{l_1}+\sigma^2)}} + e^{\frac{(\sqrt{x}+1)^2}{2(\frac{K_a-1}{l_1}+\sigma^2)}} + e^{\frac{(-\sqrt{x}+1)^2}{2(\frac{K_a-1}{l_1}+\sigma^2)}} \right] \quad (17)
\end{aligned}$$

where operation $*$ is convolution. Comparing two different distribution for each user, we can find out which distribution is more suitable. The log-likelihood function is an efficient way to identify active users. The messages are made up of independent observations. Then, the logarithm transforms a product of densities into a summation. The asymptotic properties of sums are easier to analyze.

3.3.3 Neyman-Pearson

The Neyman-Pearson lemma is part of the Neyman-Pearson theory of statistical testing, which introduced concepts like errors of the second kind, power function, and inductive behavior in [6–8]. The Neyman-Pearson lemma is a way to find out if the hypothesis test we are using is the one with the greatest statistical power. The power of a hypothesis test is the probability that the test correctly rejects the null hypothesis when the alternate hypothesis is true. The goal would be to maximize this power so that the null hypothesis is rejected as much as possible when the alternative is true. The lemma basically tells us that good hypothesis tests are likelihood ratio tests.

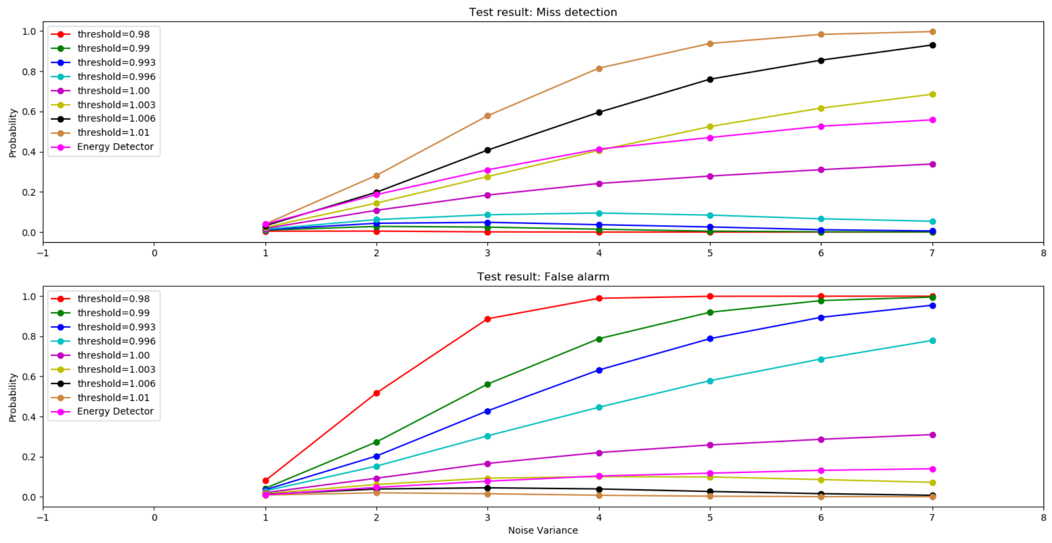


Figure 8: Different value of thershold for Neyman Pearson

Consider a test with hypotheses $H_0 : \theta = \theta_0$(inactive) and $H_1 : \theta = \theta_1$(active), where the PDF is $f(x \mid \theta_i)$ for $i = 0, 1$. Denoting the rejection region by R, the Neyman-Pearson lemma states that a most powerful test statisfies the following: for some η(threshold) ≥ 0,

$$\begin{cases} x \in R \text{ if } f(x \mid \theta_1) > \eta f(x \mid \theta_0), \\ x \in R^c \text{ if } f(x \mid \theta_0) < \eta f(x \mid \theta_1), \\ \mathbb{P}_{\theta_0}(X \in R) = \alpha \text{ for a prefixed significance level } \alpha \end{cases}$$

Define the rejection region of the null hypothesis for the Neyman-Pearson test as

$$R_{NP} = \left\{ x : \frac{\mathcal{L}(\theta_0 \mid x)}{\mathcal{L}(\theta_1 \mid x)} \leq \eta \right\}$$

Where η is chosen by designer so that $P(R_{NP} \mid \theta_0) = \alpha$ satisfy. From Figure 8, setting the difference value for the threshold we can modify miss detection and false alarm for hypothesis testing. If we set the threshold to 1.003 then we can set up as much as an energy detector. It is our choice for the device that how much willing to sacrifice false alarm to reduce miss detection.

3.4 Simulation

To be a fair comparison between schemes that introduced to the unsourced MAC and the proposed approach, we used the following parameters for numerical simulations. The number of active users $K_a = 100$ for active user detection. Each user transmits a payload B = 250 bits. These messages are encoded into n = 500 channel uses and transmitted into the channel. In active user detection, each scheme presented in this article will calculate the miss detection and false alarm like in Figure 9 and table 1. The target per-user error probability is P_e = 0.08.

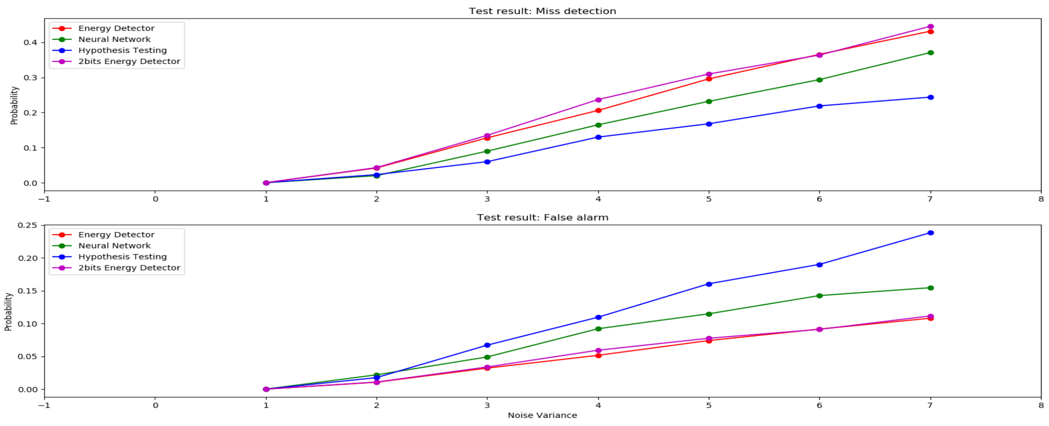


Figure 9: Miss detection and False alarm for active user detection

	Type	Noise:1	Noise:2	Noise:3	Noise:4	Noise:5	Noise:6	Noise:7
Energy Detector	Miss Detection	0	0.042	0.128	0.206	0.296	0.366	0.432
	False Alarm	0	0.0105	0.032	0.0515	0.074	0.0915	0.108
2bit Combined ED	Miss Detection	0	0.042	0.135	0.237	0.31	0.364	0.446
	False Alarm	0	0.0107	0.03375	0.05925	0.0775	0.091	0.1115
Neural Network	Miss Detection	0	0.02	0.09	0.165	0.232	0.294	0.371
	False Alarm	0	0.0217	0.049	0.092	0.1147	0.1425	0.1545
Hypothesis Testing	Miss Detection	0	0.023	0.06	0.13	0.167	0.219	0.244
	False Alarm	0	0.0175	0.067	0.1095	0.1605	0.19	0.2382

Table 1: Miss Detection and False Alarm for active user detection schemes.

4. A LOW COMPLEXITY MMSE RECEIVER

4.1 MMSE estimator

MMSE is a model that minimizes the Mean Square Error(MSE) of the received data. MMSE as an equalizer is a kind of post-processing algorithm that helps us to Figure out the received data that is close to the original signal as possible. In short, the most important step in MMSE is to find an inverse matrix for the MMSE. If we assume that there is no noise and interference, the inverse matrix can be simply an inverse of the spreading sequence matrix. But when there is noise, we need to use some model that can reflect the noise value. MMSE is one of the algorithms. In this chapter, we present the demodulation and channel decoding scheme. We denote the set of spreading sequence indices that returned by detection of an active user as D. The modulated polar codewords corresponding to all the active users are stacked into a matrix form, given by

$$V := \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}$$

We define a matrix Y, which is reshaped of the received vector $\vec{y}$, by

$$Y := \begin{bmatrix} \vec{y_1} & \vec{y_2} & \cdots & \vec{y_{n_c}} \end{bmatrix}$$

It is clear to prove that matrix Y can be denote as

$$Y = A_D V + Z \tag{18}$$

where A_D is a matrix A restricted to the columns by list D that came from active user detection. As we know, to make a simplifying approximation and matrix Z to be an independent zero-mean

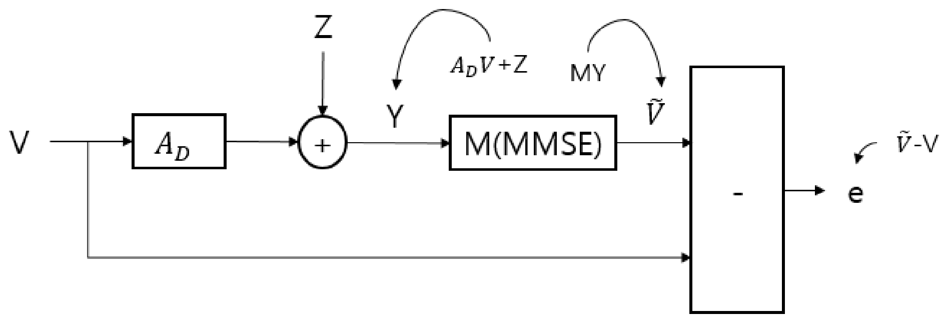


Figure 10: Block diagram of MMSE

Gaussian random variable with unit variance.

From Figure 10, we could demonstrate that the received signal and error vector has a specific condition where there is no correlation between each other. In MMSE, the matrix M shall be such a matrix that minimizes the MSE by using the statistical characteristics of the received signal. If there remains some correlation between the received vector and the error vector. the correlation should be able to utilize for decreasing the norm of the error vector. This is the reason why we are able to derive the MMSE-optimal matrix M by using the condition that claims the correlation between the received signal and the error is zero. Deriving the equation for MMSE is given by

$$\begin{aligned} E[e \cdot y^H] &= 0 \\ E[(v - My)y^H] &= 0 \\ E[vy^H] - ME[yy^H] &= 0 \\ M &= E[vy^H]E[yy^H]^{-1} \end{aligned} \tag{19}$$

Now, we have the matrix M expressed in two blocks of expectation function. Expanding each of these blocks further to derive function for MMSE is given by

$$\begin{aligned} E[vy^H] &= E[\, v(A_D v + z)^H \,] \\ &= E[\, v(v^H A_D^H + z^H) \,] \\ &= E[\, vv^H A_D^H + vz^H], \quad \text{where } E[vz^H] = 0 \\ &= E[\, vv^H] A_D^H, \quad \text{where } E[vv^H] = 1 \\ &= A_D^H \qquad\qquad (20) \end{aligned}$$

$$\begin{aligned} E[yy^H]^{-1} &= E[\, (A_D v + z)(A_D v + z)^H \,] \\ &= E[\, (A_D v + z)(v^H A_D^H + z^H) \,] \\ &= A_D E[\, v^H v] A_D^H + E[zz^H] \\ &= A_D A_D^H \,+\, \sigma^2 I \qquad (21) \end{aligned}$$

From (21), covariance matrix of vector y is denote as

$$R = (A_D A_D^T + \sigma^2 I) \tag{22}$$

We show the MMSE estimator of Y to get a linear estimate of V by equation (20) and (21)

$$M = A_D^H R^{-1}\,, \quad \hat{V} := \begin{bmatrix} \hat{v_1} \\ \hat{v_2} \\ \vdots \\ \hat{v_k} \end{bmatrix} = MY = A_D^H (A_D A_D^H + \sigma^2 I) Y \tag{23}$$

The estimated vectors corresponding to each codeword are passed into the single user decoder of the polar code. The list of decoding for polar code is successful if the codewords returned by the decoder satisfying cyclic redundancy check bits. We denote $\tilde{D}$, the list of $\vec{v_i}$ for each $i \in \tilde{D}$ that decoded successfully. In the end, we remove the contributions of all the successfully decoded codewords $V_{\tilde{D}}$ from the received signal Y to make the residual.

$$Y - A_{\tilde{D}} V_{\tilde{D}}$$

The remaining signal is repeated to do the active user detection for the second iteration. This process continues until all the transmitted signals to be successfully decoded or there is no more

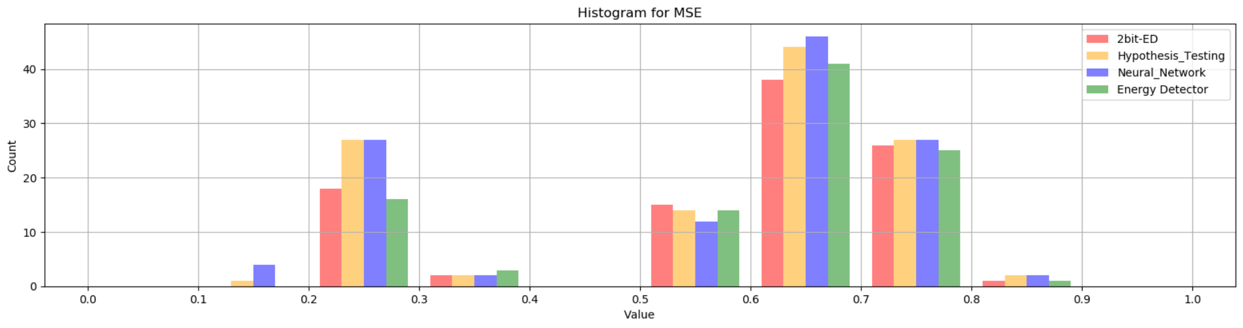


Figure 11: Histogram for MSE

that can be decoded.

4.1.1 Mean Square Error

In statistics, the MSE is an estimator that measures the average of the squares of the errors that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate. The MSE equation is given by

$$MSE = \frac{\sum_{i=1}^{B}(x_i - \hat{x_i})^2}{number\ of\ bits} \tag{24}$$

where x_i is the actual transmitted bits and $\hat{x_i}$ is the estimated MMSE bits. Like Figure 11, after the active user detection for each scheme, it shows two distinct distributions. The reason why it shows two distributions is each scheme can not detect exact active users. Each scheme has a different value of miss detection and false alarm. The left side distribution represents the inactive users because inactive users transmit with 0 value, which makes much less difference with the estimated MMSE value. On the other side, the right side of distribution is active users distribution. This has a higher difference than the inactive one. It is up to the designer to design algorithms to reduce miss detection and false alarm to get better MSE.

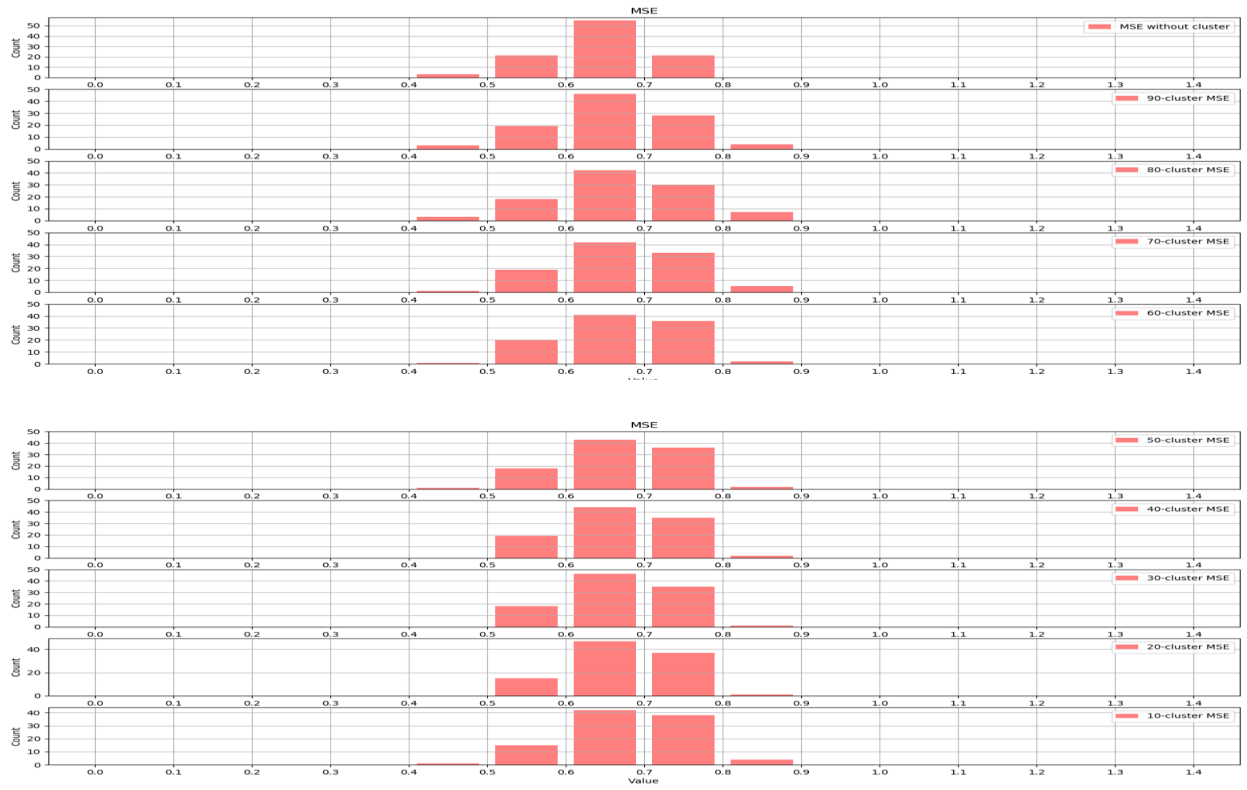


Figure 12: Histogram of MSE with different percentage of cluster size

4.2 Highly Correlated Sequences based Detector

In order to reduce dimension for the inverse matrix of MMSE, (23) can be rewritten as equivalent as following equation.

$$MY = A_D^H(A_D A_D^H + \sigma^2 I)Y \;=\; (A_D^H A_D + \sigma^2 I)A_D^H Y$$

Each user chooses fixed α highly correlated spreading sequences to reduce the matrix size for R and the remaining sequences treated as interference. Original matrix R size is $K \times K$ because $A_D^T A_D$ size is $K \times K$. But when we do clustering with α high correlated sequence, matrix size is $\alpha \times \alpha$ and the algorithms takes $\Theta(\alpha^3)$. For the clustering, we also have to calculate the interference of remained spreading sequences. we chose only α high correlated spreading sequences and $K - \alpha$ sequence are treated as interference. The sum of diagonal term for $A_{\ddot{D}}^T A_{\ddot{D}}$ will be

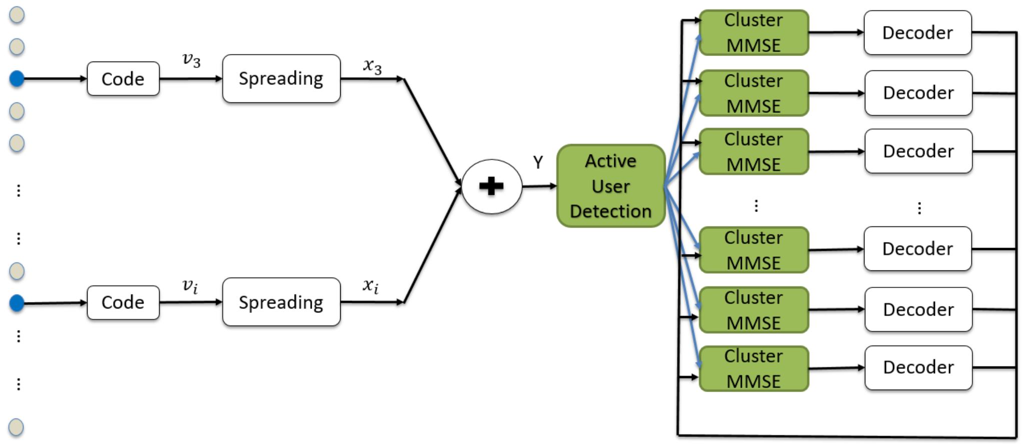


Figure 13: Block diagram for the system

add to the $\sigma^2 I$ inside of R matrix, where $\ddot{D} \in [1 : \alpha]$. After reducing the matrix size to $\alpha \times \alpha$, the interference value for each spreading sequence is the length of the spreading sequence. After clustering, we have $K - \alpha$ sequence to be treated as interference, which the value of interference is $(K - \alpha)\times$(length of spreading sequence). Figure 12 shows the distribution for different sizes of clustering α value for the histogram of MSE. It can be observed that we get a more different value between the original transmitted signal and the MMSE expected signal by setting the α value to lower. This is because setting the α value to a lower value then it will reduce the number of high correlated sequences and add more interference sequences to the inverse matrix in R.

From Figure 13, after the active user detection filter, we have the expected active user list for 4 schemes and each user inside the 4 schemes passes the individual cluster-MMSE filter to reduce the time complexity. Once again, after the cluster-MMSE filter, each user is assigned with decoder block to find out whether it is possible to decoded successfully by using the polar code. If there are users who can be decoded then using the feedback loop we remove that user from the received signal y until all the users can be decoded. reducing the matrix size is not the only way to reduced complexity. The upcoming section will demonstrate another way to reduce complexity by selecting higher chance users who can be decoded more.

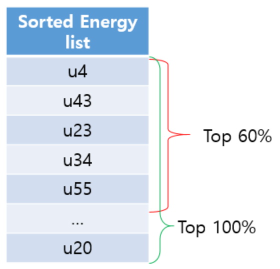


Figure 14: Example of sorted list in energy detector

4.2.1 Low complexity detector

Applications of matrix multiplication in computational problems are found in many fields including scientific computing and pattern recognition and in seemingly unrelated problems such as counting the paths through a graph in [9]. Efficient matrix multiplication would yield efficient algorithms for many other problems such as solving systems of linear equations. The author in [10] mentioned that the matrix multiplication algorithms take $\Theta(n^3)$ time to multiply two $n \times n$ matrices.

From the sorted list of the energy detector in Figure 14, We can also reduce the time complexity by selecting top-L users from the sorted list and not trying the decode all the users, because most active users are in the high rank of the sorted list. From the sorted list of active user detection, we select top-L users and try to decode them. If there are no users who can be decoded then simply move on to the next top-L users and try to decode again. This step could take more iteration to decode all the users in the set but it is possible to reduce the complexity than the original MMSE.

Another method to reduce the complexity is changing the cluster size for each user. From the sorted list of energy, we select top-L users and try to decode the first user in the list for each iteration. If that user can not be decoded then move on to the next user inside the top-L list and repeated until reaching out to the end of the list. If there are no users that can be decoded then we increase the cluster size and repeat the system. For this operation, each user will not always have the same cluster size. It is important to monitor the cluster size for each user to calculate the complexity.

		$K_a = 50$	$K_a = 75$	$K_a = 100$	$K_a = 125$
		SNR require / Complexity			
-	FD	0.8 / 1	1.2 / 1	1.4 / 1	1.9 / 1
Top:100%	Cluster: 90%	0.8 / 43.7	1.2 / 75.82	1.5 / 80	1.9 / 98.41
Top:60%	Cluster: 90%	0.8 / 26.2	1.2 / 53.05	1.5 / 64	1.9 / 75.15
Top:30%	Cluster: 90%	0.9 / 23.6	1.2 / 34.10	1.5 / 56	1.9 / 102.8
Top:10%	Cluster: 90%	0.9 / 25.3	1.2 / 40.41	1.5 / 57	1.9 / 102.8
Top:100%	Cluster: 60%	0.9 / 15.5	1.2 / 29.37	1.9 / 36	3.7 / 31.81
Top:60%	Cluster: 60%	0.9 / 10.8	1.2 / 33.04	2.0 / 29	3.7 / 19.08
Top:30%	Cluster: 60%	0.9 / 7.7	1.2 / 20.93	2.0 / 27	3.7 / 19.08
Top:10%	Cluster: 60%	0.9 / 7.7	1.2 / 22.76	2.0 / 30	3.7 / 22.79
Top:100%	Cluster: 30%	0.9 / 2.59	1.8 / 4.13	2.3 / 4	4.4 / 3.97
Top:60%	Cluster: 30%	0.9 / 1.74	2.0 / 2.47	2.4 / 5.0	4.4 / 3.37
Top:30%	Cluster: 30%	0.9 / 1.55	2.0 / 2.61	2.4 / 5.0	4.4 / 2.88
Top:10%	Cluster: 30%	0.9 / 1.75	2.0 / 2.84	2.4 / 6.1	4.5 / 3.84
Top:100%	Cluster: 10%	1.1 / 0.096	2.4 / 0.153	2.9 / 0.165	4.5 / 0.184
Top:60%	Cluster: 10%	1.1 / 0.064	2.4 / 0.091	2.9 / 0.143	5.0 / 0.125
Top:30%	Cluster: 10%	1.1 / 0.072	2.4 / 0.076	2.9 / 0.165	5.0 / 0.114
Top:10%	Cluster: 10%	1.1 / 0.075	2.4 / 0.086	2.9 / 0.188	5.0 / 0.137

Table 2: First scheme, SNR require and time complexity for each K_a value

4.3 Simulation

Table. 2 demonstrates the performance of SNR requirements and time complexity for different cluster sizes and top-L values after the MMSE estimator. For K_a=50, P_e = 0.08 and SNR of 0.8 dB is required but if we choose top-30% and cluster size 10% then we can reduce time complexity to 0.072% with 1.1 dB SNR required. For example, K_a=100, SNR of 1.4 dB is required and if we choose top-60% and cluster size 10% then time complexity is 0.143% with 2.9 dB SNR. For less K_a size, even we select a lower cluster size we can decode successfully with little difference of SNR require. But if we increase K_a size then SNR requires the same cluster size to increase dramatically. In this part, the goal is how much willingness to sacrifice SNR requires to reduce the time complexity.

Table. 3 shows the second scheme for reducing the time complexity. In this table, there are some cases that have the same complexity for different top-L values. For example, with cluster size

		$K_a = 50$	$K_a = 75$	$K_a = 100$	$K_a = 125$
		SNR require / Complexity			
-	FD	0.8 / 1	1.2 / 1	1.4 / 1	1.9 / 1
Top:100%	Cluster: 90%	0.9 / 24.8125	1.2 / 60.9268	1.5 / 90.2840	1.9 / 132.12
Top:60%	Cluster: 90%	0.9 / 24.8125	1.2 / 60.9268	1.5 / 90.2840	1.9 / 117.27
Top:30%	Cluster: 90%	0.9 / 24.8125	1.2 / 60.9268	1.5 / 96.471	1.9 / -
Top:10%	Cluster: 90%	0.9 / -	1.2 / -	1.5 / -	1.9 / -
Top:100%	Cluster: 60%	0.9 / 9.4781	1.2 / 28.6470	2.0 / 41.3007	3.7 / 31.212
Top:60%	Cluster: 60%	0.9 / 9.4781	1.2 / 29.4645	2.0 / 40.9006	3.7 / 28.805
Top:30%	Cluster: 60%	0.9 / 9.4781	1.2 / 46.1948	2.0 / 60.1950	3.7 / 34.518
Top:10%	Cluster: 60%	0.9 / -	1.2 / -	2.0 / -	3.7 / -
Top:100%	Cluster: 30%	0.9 / 2.2923	2.0 / 4.1793	2.4 / 8.3288	4.4 / 4.9235
Top:60%	Cluster: 30%	0.9 / 2.1980	2.0 / 4.1793	2.4 / 16.308	4.4 / 9.5926
Top:30%	Cluster: 30%	0.9 / 5.3754	2.0 / 11.0871	2.4 / 36.026	4.4 / 21.1746
Top:10%	Cluster: 30%	0.9 / -	2.0 / -	2.4 / -	4.5 / -
Top:100%	Cluster: 10%	1.1 / 0.0947	2.4 / 0.1525	2.9 / 0.2342	5.0 / 0.1904
Top:60%	Cluster: 10%	1.1 / 0.0947	2.4 / 0.1525	2.9 / 5.0259	5.0 / 0.6783
Top:30%	Cluster: 10%	1.1 / 3.2980	2.4 / 3.6570	2.9 / 20.4549	5.0 / 15.291
Top:10%	Cluster: 10%	1.1 / -	2.4 / -	2.9 / -	5.0 / -

Table 3: Second scheme, SNR require and time complexity for each K_a value

10% and top 100% and 60% have the same complexity. Because every time when we go down the top list, there are always users who can be decoded inside the top 60% for all users. This is why if we increase the top-L value higher than 60%, it will always get the same complexity as a top-60% case. Also with top-10%, it is unable to measure the complexity because for top-10%, there are no users who can be decoded and then we increased the cluster size 10% and repeated. But even we increase the cluster size to the original MMSE matrix size, it is still not possible to decode anyone in the top-10%. Another thing to analyze in table 3, for the same cluster size, top-100% is less or equal complexity than other cases. when we choose the top 100% then there will always be some users who can be decoded and removed from the received signal. But if we choose top-30%, there are no users who can be decoded and increase the cluster size 10% more and repeated from the first user in top-30%. This process to find the next decodable user takes more iteration to do than finding the next decodable user in top-100%.

5. CONCLUSION

We presented four activity detection schemes in chapter3 based on random spreading, single user decoding, and interference cancellation for the unsourced multiple access channel. Each scheme shows its own miss detection and false alarms. Comparison between MF and 2-bit combined MF with NN and hypothesis testing, our suggested algorithms could perform as much as the MF and 2-bit combined MF introduced in [2]. In chapter 4, we showed how to reduce the complexity based on choosing the highly correlated sequences for each user and also selecting top-L users in the expected active user list to decode in each iteration. we presented 2 methods for this process to reduce the complexity with a different approach. The spreading sequences employed in this work are random Gaussian variables. Also, the error performance of this scheme is heavily dependent on the lengths of the spreading sequence and channel code used. A goal question in this context is how could possible to identify which sequences are active with higher probability and how to reduce the complexity with a small trade-off.

REFERENCES

[1] Y. Polyanskiy, "A perspective on massive random-access," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 2523–2527, IEEE, 2017.

[2] A. K. Pradhan, V. K. Amalladinne, K. R. Narayanan, and J.-F. Chamberland, "Polar coding and random spreading for unsourced multiple access," *arXiv preprint arXiv:1911.01009*, 2019.

[3] V. K. Amalladinne, J.-F. Chamberland, and K. R. Narayanan, "A coded compressed sensing scheme for unsourced multiple access," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6509–6533, 2020.

[4] M. Elkourdi, A. Mazin, E. Balevi, and R. D. Gitlin, "Enabling aloha-noma for massive m2m communication in iot networks," *arXiv preprint arXiv:1803.09513*, 2018.

[5] K. Chitti, J. Vieira, and B. Makki, "Deep-learning based multiuser detection for noma," *arXiv preprint arXiv:2011.11752*, 2020.

[6] E. L. Lehmann, "The fisher, neyman-pearson theories of testing hypotheses: one theory or two?," *Journal of the American statistical Association*, vol. 88, no. 424, pp. 1242–1249, 1993.

[7] A. Wald *et al.*, "Chapter ii: The neyman-pearson theory of testing a statistical hypothesis," in *On the Principles of Statistical Inference*, pp. 10–20, University of Notre Dame, 1942.

[8] G. Gigerenzer, Z. Swijtink, T. Porter, L. Daston, and L. Kruger, *The empire of chance: How probability changed science and everyday life*. No. 12, Cambridge University Press, 1990.

[9] S. S. Skiena, *The algorithm design manual*. Springer International Publishing, 2020.

[10] A. N. Arslan and A. Chidri, "A clustering-based matrix multiplication algorithm," in *Proceedings of the International Conference on Scientific Computing (CSC)*, p. 1, Citeseer, 2011.