Abstract — The communication channel equalization is a difficult problem, especially when the channel is nonlinear and complex. Numerous algorithms are presented in the neural networks literature to solve this problem. In this paper, a comparison is made among the latest neural network techniques (Complex Minimal Resource Allocation Networks (CMRAN) [1]), a classical communication technique (Viterbi algorithm), and two pattern recognition techniques (Support Vector Machine (SVM), Learning Vector Quantization (LVQ)) to solve this problem. The simulation results show that Viterbi (MLSE decoding technique), and SVM methods outperform the CMRAN method.

I. INTRODUCTION

Due to thermal noise, impulse noise, cross talk, and the characteristics of the channel, the transmitting channel distorts the signals both in amplitude and phase, causing Intersymbol Interference (ISI). This distortion causes the transmitted symbols to spread and overlap over successive time intervals, causing considerable loss of information. Channel equalization is performed to minimize ISI, to restore the transmitted symbols, and to recover the signal information. To motivate the problem of channel equalization, let us consider a channel which generates the received signal samples \( y_k \) given by

\[
y_k = s_k * q_k + \eta_k = \sum_{i=0}^{L} q_i s_{k-i} + \eta_k.
\]

where \( \{q_i\} \) are channel impulse response taps of channel length \( L \), \( s_k \) are the transmitted symbols which take the values according to modulation type (QAM or BPSK etc.) used, and \( \eta_k \) is Gaussian white noise with zero mean and variance \( \sigma^2 \). The objective of an equalizer is to use the information contained in the received vector \( \hat{y}_k = [y_k \ y_{k+1} \ ... \ y_{k-d}] \) to obtain and estimate of the delayed symbol \( \hat{s}_{k-d} \) of \( s_{k-d} \).

Here, \( m \) is the equalizer order and \( d \) (\( 0 \leq d \leq m + L - 1 \)) is the equalizer delay. If the channel is linear as in (1), there are several suboptimal equalizers available in the communication literature, such as the linear minimum mean square error (MMSE) equalizer and the decision feedback equalizer (DFE). If \textit{a priori} probabilities and the statistical behavior of the channel is known, then it is easy to obtain the optimal Bayes and Maximum Likelihood (ML) solutions. These optimal techniques belong to the so-called parametric techniques, because they maximize the probability of obtaining the observed symbol (for an assumed statistical behavior of the channel), and, consequently, minimize the probability of error.

Nonparametric techniques are popular alternatives in the absence of full knowledge of the statistical behavior of the channel. Pattern recognition techniques, such as learning vector quantization (LVQ), support vector machines (SVM), and neural networks belong to the category of nonparametric methods. The channel equalization problem can be viewed as a classification problem, which involves interpreting a temporal pattern and mapping the incoming noisy signal into one of a pre-specified set of classes. The received vector is used as an input to a classifier, whose output must match the delayed version of the original signal as closely as possible in some statistical sense. SVM uses relatively small amounts of data for training, and, once the training is accomplished, the detection or testing stage is computationally efficient.

To solve the problem of nonlinear channel equalization, various neural network-based techniques have been employed in the literature. Some of the examples are multilayer feedforward networks, radial basis function (RBF) networks, and recurrent neural networks. RBF networks are presented with a set of training data containing input-output pairs, and they seek to learn the nonlinear mapping between them. Most of the neural networks used to solve the channel equalization problem employ iterative gradient-descent algorithms, which may converge to only local minima rather than global minima (optimum solution). Neural networks are also sensitive to factors such as overtraining, the number of layers, and the number of neurons per layer. They usually require long training times, and lack a systematic methodology for the network architecture selection.

Another way to solve the communication channel equalization problem is using “per packet” based detection schemes, such as the maximum likelihood sequence estimation (MLSE) using the Viterbi algorithm. We will show that Viterbi and SVM methods are more accurate than the RBF-based Complex Minimal Resource Allocation Network (CMRAN). In some cases, even the naïve LVQ is found to be better than CMRAN.

The remainder of the paper is organized as follows. We discuss the RBF-based CMRAN neural network technique in Section II. A naïve pattern recognition technique, termed the Learning Vector Quantization, is discussed in Section III. An optimal method of performing channel equalization using Viterbi algorithm is discussed in Section IV. The SVM
classifier is introduced in Section V. In Section VI, we compare the symbol error rates (SER) of various methods and their computational complexity. Finally, we conclude the paper with summary and key findings in Section VII.

II. COMPLEX MINIMAL RESOURCE ALLOCATION NETWORK (CMRAN) EQUALIZER

In the neural networks literature, various techniques are studied to solve the nonlinear equalization problem. Most of these techniques use RBF neural networks or the back propagation-based MLP method to train the neurons, i.e. to learn their weights. CMRAN is one such technique, which was recently developed to solve the channel equalization problem [1]. It is an extension of the RBF network learning algorithm developed by Yingwei et al in [2]. CMRAN is a complex version of a sequential learning algorithm, termed Minimum Resource Allocation Network (MRAN) algorithm, which employs a scheme for adding and pruning neurons, in order to achieve a parsimonious network. The response of the hidden neurons to an $m$-dimensional input vector is in the form of a Gaussian kernel. The output layer of the network is a linear combiner. The noisy output vectors from the channel are the training samples to the CMRAN equalizer, and the delayed symbols entering the channel are the desired training output. During the testing phase, the output of the CMRAN network is fed to a nearest neighbor classifier to obtain an estimate of the delayed input symbol. In order to implement complex-valued outputs, the complex-valued weights are separated into real and imaginary parts before feeding into the nearest neighbor decision device.

III. LEARNING VECTOR QUANTIZATION (LVQ) ALGORITHM

We chose to use the LVQ algorithm in this problem to represent a simplistic, naïve alternative to the channel equalization and symbol classification problem. As such, it should provide an upper bound (or worst case) performance for neural network approaches used to solve the channel equalization problem.

Consider a FIR channel of order $L$, and a symbol modulation scheme, such that a symbol can take on $n^2$ possible values. Based on this, there are $n^L$ possible inputs that result in distinct outputs from the channel. Fig. 1 illustrates this input-output mapping for a 4-QAM symbol set, where the symbol $s(k)$ can take on the values $\{-1-j, -1+j, +1-j, +1+j\}$, and the channel is of order $L=2$.

The LVQ algorithm establishes a “codebook” vector that maps the input symbol sets to their respective outputs. The symbol estimate then becomes the symbol that belongs to the codebook, and whose output is nearest (in terms of Euclidean distance) to the observed (noisy) output. Table 1 illustrates a codebook for an $L=2$ order problem using Binary Phase Shift Keying (BPSK) modulation, where symbols can take the values $\{-1, +1\}$ and channel coefficients are $\{1, 0.3\}$.

The LVQ algorithm requires estimates of the noise-free channel output for each possible input symbol sequence. To accomplish this, supervised learning takes place in the form of sending a known stream of symbol sets repeatedly through the channel, and measuring the channel output. For each input symbol set, these channel outputs are averaged, and the mean output for a given input symbol set is entered into the codebook.

The LVQ algorithm is evidently suboptimal. Specifically, it does not use all of the information in the channel output stream. For example, in an FIR channel with $L=2$, information about $s(k-L)$ is contained in the outputs $y(k)$ and $y(k-L)$. However, LVQ only uses the output $y(k-L)$ to estimate the symbol $s(k-L)$. However, as was mentioned in the introduction of this section, this implementation is intentionally simplistic, and is designed to give a “worst performance” comparison with other algorithms.

IV. MAXIMUM LIKELIHOOD SEQUENCE ESTIMATION (MLSE—VITERBI ALGORITHM)

A maximum likelihood sequence estimator (MLSE) is well known method of performing channel equalization. Given a sequence of symbols of length $N_{sym}$, the MLSE provides the most likely symbol sequence, given the observed channel output. This is distinct from symbol-by-symbol classifiers, which classify a symbol based on a single (or vector of size equal to the length of the FIR channel) channel output measurement. The optimal symbol-by-symbol classifier identifies the symbol that maximizes, for an equalizer order of $m$, and delay $d$, the pdf $f(s(k-d)|y(k),y(k-1),...,y(k-m+1))$. However the MLSE classifier uses additional information in the pdf construction, namely, the optimum symbol estimates that surround the symbol being classified:

$$f(s(k-d)|y(k),y(k-1),...,y(k-m+1),s(k),s(k-1),...,s(k-m+1),s(k-m-1),...,s(k-m+L-1)).$$

Given this additional information about the
optimum estimate of the remaining symbols in the symbol sequence, the MLSE can classify the entire symbol sequence more accurately than the optimal symbol-by-symbol classifier. The Viterbi algorithm represents an efficient means of performing MLSE. The algorithm uses dynamic programming, and requires that the symbol sequence be preceded and terminated by known symbols (of length $L-1$, where $L$ is the FIR channel length). Fig. 2 illustrates the Viterbi algorithm on a BPSK symbol set and an FIR channel of length $L=2$.

![Viterbi Algorithm Illustration](image.png)

The first part of the Viterbi algorithm is actually similar to the LVQ algorithm in that, for a given input symbol set, the Euclidean distance between the observed output and the noise-free codebook output is computed. As the Viterbi processes the measurements forward, the cumulative distance at each node of the Viterbi trellis is computed. Based on the cumulative distance of a path arriving at a node, the path with the lowest cost is retained. That is, at each node, the node retains the cumulative cost to arrive at that node, and the optimum preceding node that yields this minimum cumulative cost. Once the algorithm passes forward through the measured data, it then passes backwards and extracts the optimum (MLSE) path, or the sequence of symbols.

Use of the Viterbi algorithm is not restricted to linear channels, but can operate on non-linear channels as well. Additionally, if the channel is not known, the channel output response can be estimated using training symbol sequences similar to that done in the LVQ algorithm. In the Viterbi algorithm implementation presented in this paper, the symbol sequence is of length 1,000 wrapped in the known pre- and post-packet symbols.

V. SUPPORT VECTOR MACHINES (SVM)

Support vector machines are one of the best classification methods available. The SVM was first introduced in [6], [7] and is widely applied in areas such as handwritten digit recognition, anomaly detection in computers, text classification, fault diagnosis, etc. The key to SVM is one of finding a hyperplane that maximizes the separation between classes. SVM uses nonlinear pre-processing techniques (“kernel”) to project the data from a low dimensional space (input space) to a high dimensional space (feature space). The linear operation in the feature space is equivalent to non-linear operation in the input space. It finds an optimal hyperplane in the high dimensional space using quadratic programming. The reader is referred to [7] for details on the SVM classifier.

The SVM uses “kernel trick” to overcome the costly computations in the high dimensional feature space. The “kernel trick” refers to representing the data, which appears as an inner product in the feature space, using a positive definite kernel function. The selection of kernel is one of the key steps in the SVM algorithm. There are no direct guidelines for the selection of kernel functions; however a RBF kernel is known to perform well in most applications.

The procedure involved in using the SVM is similar to those of other classification methods. It uses training patterns to learn alpha values (Lagrange multipliers associated with the constraints of the optimization problem) using quadratic programming, and finds support vectors (non-zero alpha values) to determine the optimal hyperplane. We need to specify the kernel parameter, $\gamma$ and cost relaxation parameter $C$. The selection of these parameters is an active research area. In this paper, these parameters are empirically computed. We have used $\gamma=0.5$, and $C=4$ in all the simulations.

Fig. 3 shows the SVM model of equalization during the training process. An input sequence consisting of QAM modulated symbols $s(k) \in \{\pm 1 \pm j\}$ is fed into the noisy nonlinear channel. If the channel is complex, then the output of the channel, $y(k)$, is separated into real and imaginary parts and fed to the SVM as features of the training pattern. Each training pattern to the SVM consists of $2^m$ features, where $m$ is the equalizer length. A delayed version of the input sequence $s(k-d)$ is fed to the SVM equalizer as training targets. SVM uses the training data to learn the support vectors and the alpha values, which are used during the testing phase.

The equalizer model of SVM during the testing phase is shown in Fig. 4. The nonlinear channel outputs are separated into real and imaginary parts, and they are delayed by a feedforward delay of length $m-1$. The SVM equalizer uses the support vectors and alpha values (which were learned during the training) to estimate a delayed input symbol $\hat{s}(k-d)$. A detailed description of the SVM equalizer is presented in [8].
VI. RESULTS

To compare the performance of various equalizers, we performed simulations for all the examples presented in [1]. Due to space constraints, we are presenting only three examples here.

Example 1: Real Linear Channel (Patra’s Model [3])

The normalized transfer function of the channel is given by (2)

\[ c(z) = \frac{o(z)}{s(z)} = 0.447 + 0.894z^{-1} \]  

We are using a linear channel i.e. \( y(k) = o(k) \). The channel order is \( L=2 \). The input symbols are generated for 4-QAM modulation, in which the real and imaginary parts are independent and uniformly distributed. The input signals are from the set \{+1,-1\}. The signal power is assumed to be unity. The output sequence is generated by mixing the channel output with additive white Gaussian noise (AWGN) to obtain the desired SNR. The equalizer order and decision delay are set to one. The CMRAN and SVM equalizers are trained on 3000 training samples and tested on 10^5 samples. The Viterbi algorithm is run for 1000 symbol packets. Fig. 5 shows the Symbol Error Rate (SER) performances of various equalizers. Evidently, Viterbi and SVM perform better than the best neural network equalizer i.e. CMRAN.

Example 2: Complex Linear Channel (Chen’s Model [4])

The normalized transfer function of the channel is given as

\[ c(z) = (0.7409 - j0.7406)(1 - (0.2 - j0.1)z^{-1})(1 - (0.6 - j0.3)z^{-1}) \]

The channel is linear and it is of order \( L=3 \). The real and imaginary parts of the input sequence are selected from the set \{+1,-1\}. The model consists of 64 states in the channel as shown in Fig. 6. The signal power is unity. We considered equalizer order and decision delay as 1 for this model. The CMRAN and SVM equalizers use 1200 samples for training and 10^5 testing samples. The LVQ and Viterbi algorithms are run for 64 states. Fig. 7 shows that the SVM and Viterbi equalizers outperform the CMRAN equalizer. In addition, the naive LVQ algorithm also outperforms CMRAN.

Example 3: Complex Nonlinear Channel (Cha and Kassam’s Model [5])

In this example, a highly complex nonlinear channel is used to compare the SER performance of various equalizers. The noisy channel output \( y(k) \) is given as

\[ y(k) = o(k) + 0.1o(k)^2 + 0.05o(k)^3 + v(k) \]
\[ v(k) \sim N(0, 0.01) \]
\[ o(k) = (0.34 - j0.27)s(k) + (0.87 + j0.43)s(k - 1) \]
\[ + (0.34 - j0.21)s(k - 2) \]

The CMRAN equalizer is of order three and has a decision delay of one. The real and imaginary parts of the input sequence are selected from the set \{+0.7,-0.7\}. The channel has 64 states, and signal power is assumed to be unity in simulations. The SVM equalizer order and delay are set as 1. Fig. 8 shows that the SER performance of the SVM and Viterbi equalizers is better than that of the CMRAN at all SNRs.
network techniques (MLP with RBF leading to the CMRAN implementation) with other pattern classification techniques (LVQ and SVM) as well as other methods (MLSE using the Viterbi algorithm implementation) is in order.

Our study has demonstrated that the best MLP-based technique from the literature (CMRAN) did not perform as well as SVM. Indeed, CMRAN performed worse on one of the three test cases than a simple LVQ implementation. Further, CMRAN was computationally more expensive than other techniques. Thus, from the viewpoint of a variety of metrics, MLP and RBF-based techniques for the channel equalization problem are inferior.

A second aspect of this study analyzed the performance of pattern classification algorithms to the MLSE (Viterbi) algorithm. The pattern classification techniques investigated here (CMRAN, LVQ, SVM) operate on a symbol-by-symbol classification system. The MLSE performs classification on packets of symbols. This packet-based approach allows for improved performance over symbol-by-symbol classifiers. In fact, for the same Symbol Error Rate (SER), the MLSE algorithm can operate at SNR levels 10-15 dB lower than the symbol-by-symbol classifiers. This translates into a 10 to 40 fold decrease in transmitted power. This significant reduction in required transmit power will generally drive the communications system design towards the packetized format required by the MLSE.

As a result, our conclusion is that the pattern classification approaches studied here, and by extension other symbol-by-symbol pattern classification techniques from the literature, will generally not be considered in communications system design.

REFERENCES