ASC Major Project

Neural Network Based Sequence Detection Over Molecular Mommunication Channels

Peter Hartig

Lehrstuhl für Digitale Übertragung
Prof. Dr.-Ing. Robert Schober
Universität Erlangen-Nürnberg

Betreuer: Prof. Dr.-Ing. Robert Schober
Prof. Dr.-Ing. Wolfgang Gerstacker

March 4, 2020

idc

FAU FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Declaration

To the best of my knowledge and belief this work was prepared without aid from any other sources except where indicated. Any reference to material previously published by any other person has been duly acknowledged. This work contains no material which has been submitted or accepted for the award of any other degree in any institution.

Erlangen, March 4, 2020

Siglinde Musterstudi
Musterstrasse 1
Erlangen

Contents

Abstract

Estimation of the probability distribution function characterizing a communication channel is investigated. In this work, pilot symbol sequences are used to train a neural network to estimate unknown channel probability distribution functions. The estimated function is then evaluated as a metric for the Viterbi algorithm. In particular, the detection performance is investigated for channels relevant to the molecular communications domain. A proposal is then made to reduce the complexity of the resulting detection scheme using a second supervised learning method. The neural network based estimation is tested in simulations and shown to be effective in detection for (DESCRIBE CHANNELS) channels.

Glossary

Abbreviations

Operators

$(\cdot)^*$	Complex Conjugate
$(\cdot)^{-1}$	Matrix Inverse
$(\cdot)^T$	Matrix Transpose
$(\cdot)^H$	Matrix Hermetian (Matrix Conjugate Transpose)
$\|\cdot\|$	Euclidean Norm
$E\{\cdot\}$	Expectation
$\text{sign}(\cdot)$	Sign Operator
$\text{floor}(\cdot)$	Floor Operator

Notation

The following notation is used through the following work.

$p(x)$ is the probability of an event x. $p(x|y)$ is the conditional probability of x given y. $E[x]$ is the expected value of a random variable . $x \sim \mathcal{N}(0,1)$ denotes a random variable x whose probability distribution is Gaussian with mean 0 and variance 1.

Vectors are denoted by bold font lower-case letters ($\mathbf{x}$) and are assumed column vectors. The vector $\mathbf{x}_i^j$ denotes a vector containing the elements i through j of $\mathbf{x}$. Sets are denoted using upper-case calligraphic letters ($\mathcal{A}$) and the cardinality of a set $\mathcal{A}$ is denoted by $|\mathcal{A}|$. $\underset{x}{\text{argmin}} f(x)$ is the value of variable x which minimizes the function $f(x)$.

Chapter 1

Report

1.1 Introduction

The general communication channel is equivalent to a conditional probability distribution function (pdf). The channel pdf $p(\mathbf{x}|\mathbf{y})$ takes into account the potentially random channel though which the transmitted information $\mathbf{x}$ passes to reach a receiver as the information $\mathbf{y}$ [?, Ch. 7]. Selecting the most probable transmitted information sequence based on the received information sequence is known as maximum a posteriori (MAP) sequence estimation. In general, sub-optimal solutions to this estimation problem do not require perfect knowledge of the pdf $p(\mathbf{x}|\mathbf{y})$ and may be used when the true $p(\mathbf{x}|\mathbf{y})$ is unknown or impractical to obtain.

Some communication contexts, including wireless, have successfully used "pilot" symbol-streams to estimate the pdf $p(\mathbf{x}|\mathbf{y})$ [?]. The receiver is informed of a symbol sequence prior to receiving that same sequence sent through the channel. The receiver may then compare the received signal with the known ground truth to estimate the channel $p(\mathbf{x}|\mathbf{y})$. The use of pilot sequences typically relies on a pre-selected channel model whose parameters are determined using the pilot sequence. For example, if the channel is assumed to be a noiseless, linear and time-invariant (LTI) system, the pilot sequences may be used to find the exact impulse response of the channel as will be seen in this work. In some cases, however, a precise channel model may not be available. This work investigates the parameterization of a very general channel model for $p(\mathbf{x}|\mathbf{y})$ using a neural network. In previous work, a neural network-based estimation of the channel has shown to be an effective method for detection with incomplete channel information for linear, time-invariant channels [?] [?]. Extending on this, we look to estimate $p(\mathbf{x}|\mathbf{y})$ for channels in the molecular communication domain. We also look to exploit any redundancy in the pdf $p(\mathbf{x}|\mathbf{y})$ in order to reduce the complexity of the resulting detection algorithm.

Molecular communication channels can be difficult to characterize due to their non-linear and random components that may or may not depend upon the transmitted information. As a result, molecular communication channels are good candidates for testing this estimation technique. For a thorough review and mathematical formulation of molecular communication channels and their applications, see [**?**].

1.2 System Model

A system model is now developed as a consistent communication framework to be used in the remainder.

We consider a point-to-point communication system as in Figure 1.1 with a transmitter sending a sequence of information **x** over a channel. At the receiver, the sequence **y** is detected. The elements of **x** are chosen from a finite symbol alphabet $\mathscr{A}$ such that $x[\text{k}] \in \mathscr{A}$.

Figure 1.1: The point-to-point communication system.

The components of the vector **y** are described by

$$y[\text{k}] = f_\text{k}(\mathbf{x}) + n[\text{k}],$$

with independent $n[\text{k}] \sim \mathscr{N}(0,1)$. Allowing for $f_\text{k}()$ to be a general and potentially random function, each received symbol $y[\text{k}]$ is potentially a function of all input information **x**.

By assuming all $n[\text{k}]$ to be independent, an orthogonal filtering of the modulated information **x** at the receiver is implied. We define the signal to noise ratio (SNR) at the receiver as

$$\text{SNR} = \frac{E\{|x[\text{k}]|^2\}}{\sigma^2}.$$

We now consider specific cases for the function $f_\text{k}()$ that will later be used in describing the detection methods.

In the first communication channel $f_k(\mathbf{x}) + n[k]$ considered, each $y[k]$ is a causal, linear, and time-invariant (LTI) combination of the transmitted sequence $[x[k], x[k-1]...x[K-L+1]]$ weighted by coefficients $[a[0], a[1]...[L-1]]$.

$$y[k] = \sum_{l=0}^{L-1} a[l]x[k-l]. \tag{1.1}$$

We also consider the case in which the received output from an arbitrary communication channel is quantized giving

$$y[k] = \text{Quant}(f_k(\mathbf{x}) + n[k]) \tag{1.2}$$

with

$$\text{Quant}(x) = \begin{cases} \text{sign}(x)\text{floor}(\frac{x}{\text{step size}}) & \text{lower saturation level} \leq x \leq \text{upper saturation level} \\ \text{upper saturation level} & x \geq \text{upper saturation level} \\ \text{lower saturation level} & x \leq \text{lower saturation level} \end{cases}$$

for

$$\text{step size} = \text{floor}\left(\frac{|\text{upper saturation level} - \text{lower saturation level}|}{\#\ \text{Quantization levels}}\right).$$

A quantization function with 20 quantization levels and lower/upper saturation levels -5/+5 respectively is shown in Figure 1.2.
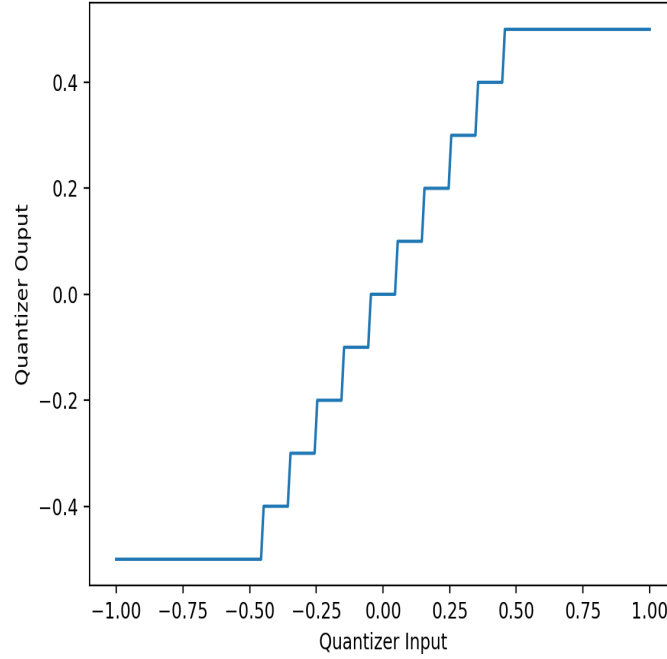
Figure 1.2: A Quantization function with lower saturation level -0.5 and upper saturation
        level 0.5.

1.3 Background

The following section first introduces the optimization problem used in the remainder of
this work. An efficient implementation for solving the optimization problem is then intro-
duced. With this foundation in place, the data-driven, neural network-based estimator is
then incorporated into the optimization algorithm. Extending on the neural-network-
based detector, a method to further reduce the algorithm complexity is then considered.
Last, a linear equalizer is derived to be used as a reference in the simulation results.

1.3.1 The Viterbi Algorithm

Rather than considering only the transmitted symbols $\mathbf{x}$ for the MAP problem described
above, we consider instead a system in which a sequence $\mathbf{y}$ is received from a channel
which may take on a set of states $\{s_1, s_2 ... s_L\} = \mathscr{S}$. With knowledge of the potential
states $\{s_1, s_2 ... s_L\}$ but not the sequence of realized states $[s[1], s[2] ... s[N]]$ corresponding
to $[y[1], y[2] ... y[N]]$, the MAP problem is restated as

$$\operatorname*{argmax}_{\mathbf{s} \in \mathscr{S}^N} p(\mathbf{s}|\mathbf{y}).$$

Using Bayes' theorem, we can equivalently use

$$p(\mathbf{s}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{s})p(\mathbf{s})}{p(\mathbf{y})}.$$

The term $p(\mathbf{y})$ is independent of $\mathbf{s}$ so it is removed without effect. The resulting optimization problem

$$\underset{\mathbf{s}\in\mathscr{S}^N}{\operatorname{argmax}} \; p(\mathbf{y}|\mathbf{s})p(\mathbf{s}) \tag{1.3}$$

will be used in the remainder. Noting that the size of the search space $|\mathscr{S}^N|$ grows exponentially in N, the number of received symbols, a method for reducing the search complexity is now introduced.

For the LTI communication channel described in the system model section, each state $s_i \in \{s_1, s_2...s_L\}$ represents a possible transmitted sequence $[x[i-L+1]...x[i]]$ and $p(\mathbf{x}) = p(\mathbf{s})$. If all symbols in $\mathscr{A}$ are selected with equal probability, as is commonly assumed to maximize information entropy for finite $|\mathscr{A}|$, the term $p(\mathbf{x})$ can be removed from problem 1.3 without effect. For this channel,

$$\underset{\mathbf{s}\in\mathscr{S}^N}{\operatorname{argmax}} \; p(\mathbf{y}|\mathbf{s})p(\mathbf{s}) = \underset{\mathbf{s}\in\mathscr{S}^N}{\operatorname{argmin}} \sum_{i=1}^{N} -\log(p(y[i]|s[i])p(s[i]|s[i-1])) =$$

$$\underset{\mathbf{x}\in\mathscr{A}^N}{\operatorname{argmin}} \sum_{i=1}^{N} -\log(p(y[i]|[x[i-L+1]...x[i]])p([x[i-L+1]...x[i]]|[x[i-L]...x[i-1]]))$$

in which individual terms of the sums are statistically independent. The optimization problem is equivalently represented by a graph with the L states $\{s_1, s_2...s_L\}$ repeated N times and all states in time k, $s_i[k]$, $i = 1... L$ connected to all states in time k+1, $s_i[k+1]$, $i = 1... L$ by an edge weighted with

$$-\log(p(y[k+1]|s_i[k+1])p(s_i[k+1]|s_j[k])).$$

This is illustrated in the following example.

We consider the communication system with $\mathscr{A} = \{0, 1\}$ and received symbols

$$y[k] = a[0]x[k] + a[1]x[k-1] + n[k],$$

with $n[k] \sim \mathscr{N}(0, 1)$. The resulting state graph (or trellis) is depicted in Figure 1.3.
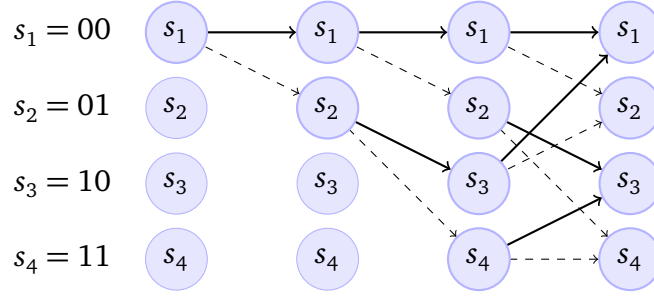
Figure 1.3: MAP decoding for and LTI channel with $\mathscr{A} = \{0,1\}$ and channel memory $L = 2$. Each state $s_1 ... s_4$ represents a possible transmit symbol sequence $[x[i-1], x[i]]$.

Again assuming equiprobable $x[k] \in \mathscr{A}$, the value of $p(s_j[k+1]|s_i[k])$ for connected states $s_j[k+1]$ and $s_j[k]$ is 0, if the channel memory represented by $s_j[k+1]$ and $s_j[k]$ contradicts. $p(s_j[k+1]|s_i[k])$ is some constant c ($0 < c \leq 1$) if these states agree on the channel memory. For clarity, all edges for which $p(s_j[k+1]|s_i[k]) = 0$ have been removed in Figure 1.3. Therefore the edges into state $s_i[k+1]$ are weighted by $-\log(p(y[k+1]|s_j[k+1])$. The solution to problem 1.3 corresponds to the minimum cost path through all N sets of states which can be found using the Viteribi algorithm.

Viterbi Algorithm

given $p(y[k]|s_i[k])$, $p(s_l[k+1]|s_j[k]) \; \forall \; k \in 1..N$ and $\forall \; j \in 1..L$.

for $k = 1...N$

    for $i = 1...L$

        Let $\text{cost}_i[k] = \underset{j \in \{1...L\}}{\text{argmin}} \left(\text{cost}_j[\text{k-1}] - \log(p(y[k]|s_i[k])p(s_i[k]|s_j[\text{k-1}])) \right)$

return detected transmission $\hat{\mathbf{x}}$ corresponding to trellis path of $\underset{i \in \{1...L\}}{\text{argmin}} \; \text{cost}_i[N]$

Note that the Viterbi algorithm is *linearly* complex in the length of the sequence $\mathbf{s}$ (N in the algorithm above) but *exponentially* complex in the number of channel states $|\mathscr{S}|$.

The above model allows channels whose state is based upon transmitted symbols to be represented using a trellis, such as the LTI channel. This graph-based model can also generalize to represent channels with certain kinds randomness that is *independent* of the transmitted symbols. In this case, the model is constrained to channels whose state satisfies the property

$$p(\mathbf{s}) = p(s[\text{N}]|s[\text{N-1}])p(s[\text{N-1}]|s[\text{N-2}])...p(s[2]|s[1])p(s[1]),$$

which implies the Markov property

$$p(s[k] | [s[1]...s[k-1]], [s[k+1]...s[N]]) = p(s[k]|s[k-1]).$$

An example of such a channel is the same LTI system described above, but with a two-state noise floor satisfying the Markov property (TODO simulate and if results then detail model and reference to results)

Finally, this model can also represent the case in which a channel state $s[k]$ does not uniquely identify a symbol $x[k]$ that was transmitted in the corresponding time index. While this case requires additional tools to ultimately arrive at a unique transmit sequence **x** it may also reduce the complexity of the corresponding Viterbi algorithm. This case will be considered in more detail later in this work.

In the remainder of this work, we consider the case in which the channel state is determined entirely by $[x[i-L+1]...x[i]]$, as was the case for the basic LTI channel (Remove if results above work).

1.3.2 ViterbiNet

Despite the complexity reduction of the Viterbi Algorithm for MAP detection, weighting the edges of the graph requires knowledge of the channel probability distribution $p(y[k]|s[k])$. Rather than estimate this function directly, we use Bayes' theorem and instead estimate the individual terms of

$$p(y[k]|s_i[k]) = \frac{p(s_i[k]|y[k])p(y[k])}{p(s_i[k])}.$$

- $p(s_i[k]|y[k])$: For a channel with finite states $\{s_1, s_2, ..s_L\}$, this probability mass function can be estimated using a neural network for classification. Each training data pair includes a single received pilot symbol $y[k]$ (the network input) and the probability of being in a state $s_i[k] \sim [x[i-L+1]...x[i]]$ (the network output). The true state is known for pilot symbols. By using using a network architecture with a number of outputs corresponding to the number of channel states, the trained network will output the desired $p(s_i[k]|y[k])$ for all $s_i \in S$. Figure 1.4 depicts the neural network for a channel with four states. The shown network has a total of 20 weighted edges (model parameters) to be optimized during training. For a detailed review of classification using neural networks, see [**?**, Ch. 5].

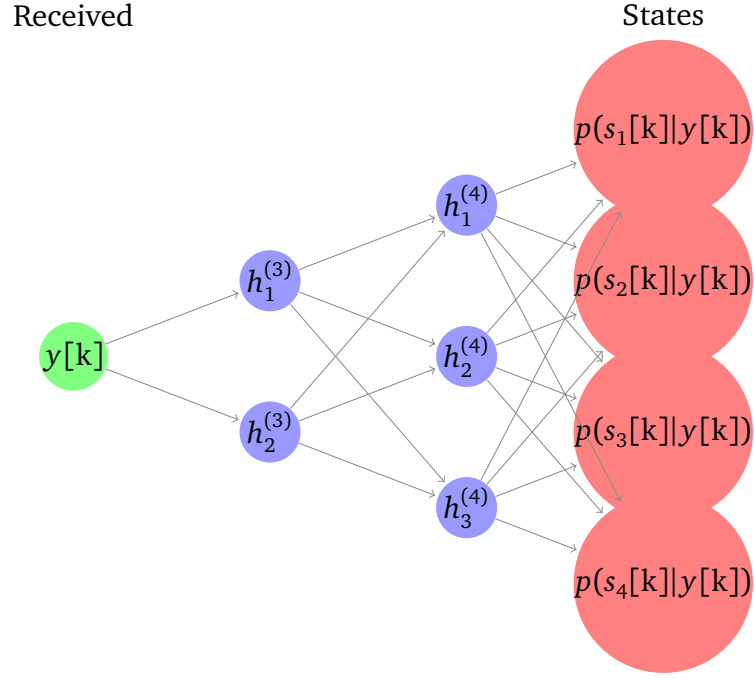Received   States

Figure 1.4: A fully-connected, feed-forward neural network architecture for classification of received symbols $y[k]$ into four channel states.

- $p(y[k]) = \sum_{s_i \in \mathscr{S}} p(s_i, y[i])$: This term is found by marginalizing the joint probability of channel state $s_i[k]$ and received signal $y[k]$ over all channel states $\mathscr{S}$. As $p(y[k])$ is constant over all states in a given step of the Viterbi algorithm, it may be interpreted as a weighting factor for the cost of received symbol $y[k]$ relative to other symbols $y[j]$, $j \neq k$. For example, if $y[k]$ is received under improbable noise and channel state, this term would reduce the overall impact of $y[k]$ on the final path chosen through the graph. In the case of equiprobable channel states without receiver noise $p(y[k])$ would be constant and have no impact on the chosen path.

This probability distribution function can be estimated using a so-called mixture model. This requires pre-selecting the number of considered channel states and a parameterized model for receiver noise. The Expectation Maximization algorithm, detailed in [**?**], along with a set of channel output to train the model with, can be used to optimize this model. Unlike parameterizing the neural network this algorithm does not require labeled data. An example of a Guassian mixture model calculated for the output of an LTI channel with 16 states (Gaussian sources) is shown in Figure 1.5.

Figure 1.5: A mixture of complex, Gaussian sources and the averages of the Gaussian sources predicted using the Expectation Maximization algorithm

- $p(s[k])$: For the case in which each channel state is equiprobable, such as when s_k corresponds to $[x[i-L+1]...x[i]] \in \mathscr{A}_{i-L+1}^i$, this term will not influence the final path and can be neglected.

Combining these terms and their impact on the original $p(y[k]|s_i[k])$, each edge weight used in the Viterbi algorithm essentially gives the probability of simultaneously arriving at a state $s_i[k]$ conditioned on the previous state $s_i[k-1]$ and the probability of receiving $y[k]$ based on the channel and noise. We now consider how a channel might be represented using the minimum number of states.

1.3.3 A Reduced-State ViterbiNet

While the Viterbi Algorithm complexity scales exponentially in the number of states possible in each step of the algorithm, in some cases this complexity can be reduced without significant loss of detection performance. One well-established method to achieve this is by pre-filtering the incoming signal so as have a minimum-phase system representing the channel (TODO Cite). Such methods, however, are often dependent upon an estimate of the channel, again implying an assumed underlying channel model. In this section, we consider a method for this complexity reduction without pre-filtering the estimated channel.

There are two things to note here, first, this pre-filtering works for LTI channels but may not be possible for channels with non-linearities (such as molecular communication channels). Also note that even this minimum phase representation of the channel still has redundancy that could be reduced using this method. Finally, this pre-filtering assumes some sort of estimate and model of the channel that we are trying to abstract from in this work for generality.

In the following, relevant example a channel is proposed whose states, as seen by the receiver, contain redundancy. One method for reducing the number of states based on this observed redundancy is then considered.

The received signal from an LTI communication channel

$$y[k] = \sum_{l=1}^{L} a[l]x[k-l] + n[k], \; n[k] \sim \mathscr{N}(0,1),$$

with $\mathscr{A} = \{-1, 1\}$, $n[k] \sim \mathscr{N}(0, 1)$ and

channel impulse response $\mathbf{a} = [a[1]...a[5]] = [1, 0, .2, .2, .4]$ ($\|\mathbf{a}\|_2^2 = 1$), is shown in Figure **??**. Desipite having channel memory L = 5, there are fewer than the potential $|\mathscr{A}_{i-L+1}^i| = 2^5$ clusters of received signals. This is due to the redundancy in the channel. As one of the channel taps is 0, this will have no impact on the output seen at the receiver

and thus 16 of the potential 32 states are removed. Further, as there are two taps of value 0.2 and $\mathscr{A} = \{-1, 1\}$, when $[x[k-2], x[k-3]] = [-1, 1]$ and $[x[k-2], x[k-3]] = [1, -1]$ these map to the same output as seen by the receiver.

One data-driven approach to exploiting state redundancy is to cluster the observed states (receiver output) into a set of $k < |\mathscr{S}|$ clusters using training data. The k-Means algorithm is proposed as one such clustering method.

---

k-Means Clustering Algorithm: A method for unsupervised classification

---

given initial location L_c, $\forall\, c \in \{1..K\}$ of K centroids.
for Number of Iterations.
    for each training data point x_i, $i \in \{1..N\}$
        1. Label x_i with the index c of the closest centroid using $\|x_i - L_c\|_2^2$.
    for centroid L_c, $\forall\, c \in \{1..K\}$
        1. Move L_c to the average location of all training points with label .c
return Centroid locations.

An important implementation consideration for this clustering approach is now discussed. If the channel has memory $L > 1$, at least some of the channels states represent different channel inputs as described previously. If states with contradicting transmission memory are clustered during the state reduction, the choice for the transmitted symbol is unclear. To accommodate this scenario, a majority rule decision is made, based on the final state path chosen by the Viterbi algorithm. During the clustering, the fraction of training samples from the original states labeled in each reduced state are recorded to give the likelihood of each symbol in the state memory. After finding the lowest cost path through the trellis, the probability of a symbol for each time index is found and the most likely is chosen. Note that this operation is *not* exponentially complex in the memory length of the channel.

Clearly an appropriate number of clusters will influence the performance of the resulting decoder (Have figure showing this).

1.3.4 Equalization benchmark

The Linear Minimum Mean-Squared Error (LMMSE) equalizer derived below is used provide a linear equalization reference to the non-linearity of the proposed neural network-based equalization .

Given an estimate of channel memory length L, the convex and continuously differentiable optimization problem

$$\underset{\mathbf{h}\in\mathscr{C}^{L}}{\text{argmin}}\; E[\|x - \mathbf{h}^T\mathbf{y}\|^2],$$

minimizes the squared difference of the linearly detected symbol $\hat{x} = \mathbf{h}^T\mathbf{y}$ and true symbol x. Using

$$\frac{\partial E[\|x - \mathbf{h}^T\mathbf{y}\|^2]}{\partial \mathbf{h}} = 0$$

the optimal LMMSE equalizer is [?]

$$\mathbf{h} = E[\mathbf{R}_{yy}]^{-1} E[\mathbf{r}_{yx}].$$

$E[\mathbf{R}_{yy}]$ and $E[\mathbf{r}_{yx}]$ are estimated by

$$E[\mathbf{R}_{yy}] = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}^i_{\text{i-L+1}} {\mathbf{y}^i_{\text{i-L+1}}}^H$$

and

$$E[\mathbf{r}_{yx}] = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}^i_{\text{i-L+1}} x_{\text{i}},$$

using the same training data as the neural network and mixture model.

One interesting note to make in comparing this equalizer with the neural network-based equalizer is the amount of training data needed. In the case of the noiseless LTI system, a linear estimator for the channel requires only L training examples to form a deterministic system of equations and learn the channel perfectly. In contrast the neural network will generally have many edge weights and would therefore require more training data. The need for a larger training data set comes with the added potential to learn more complicated channels with the neural network than with the LMMSE equalizer.

In the results below, the theoretical symbol error rate curve for the memoryless, AWGN channel is also included as a reference. For all of the tests below, an anti-podal BPSK $\mathscr{A} = \{-1, +1\}$ scheme is simulated giving [?]

$$p(\text{symbol error}) = \frac{1}{\sqrt{\pi \sigma_{\text{Noise}}}} \int_{-\infty}^{0} exp\left[-\frac{(r - \sqrt{E_b})^2}{\sigma_{\text{Noise}}}\right]$$

with r as the received signal and E_b being the energy per bit at the receiver.

1.4 Results

1.4.1 Simulation Details

The following simulations are performed using only real-valued transmit symbols, channels and noise. As a result, an SNR adjustment factor of 3dB has been used in order to align with well-know results from other sources.

1.4.1.1 Implementation Details

This section defines relevant information for implementation of the simulation used to generate the results below.

- **Neural Network** The neural network used in this simulation was implemented using the PyTorch machine learning library. The network architecture and training parameters used are as follow:

  - Architecture: 4 layers 1, 100 (Tanh activation), 50 (relu activation), M^L (Softmax → Negative Log-Likelihood)

  - Training Data Examples: 5000

  - Neural Network Updates: Adam [**?**] with step size 10^{-2}

  - Batch Size: 1000

  - Backpropogation Updates (Epochs): 900

  - Loss Function: Cross Entropy

- **Expectation Maximization Algorithm**

  - Training Data Examples: 5000

  - Mixture Model Source Types: Gaussian

  - Number of Sources: The number of channel states for specific simulation

- **k-Means Algorithm (For Reduced State ViterbiNet)**

  - Training Data Examples: 5000

- **Linear Minimum Mean-Squared Error (LMMSE) equalizer**

  - Training Data Examples: 5000

  - Equalizer Channel Memory: Always provided with the true channel memory length.

1.4.2 Simulation Results

The following section evaluates the performance of the equalizers described in this work using simulations

We first evaluate the detector performance using linear, time-tnvariant channels as defined in equation (1.1). In order to compare the performance of the reduced state equalizer, we consider cases with channel redundancy (Fig. 1.8) and without channel redundancy (Fig. 1.9). with impulse response $a[l] = ...$
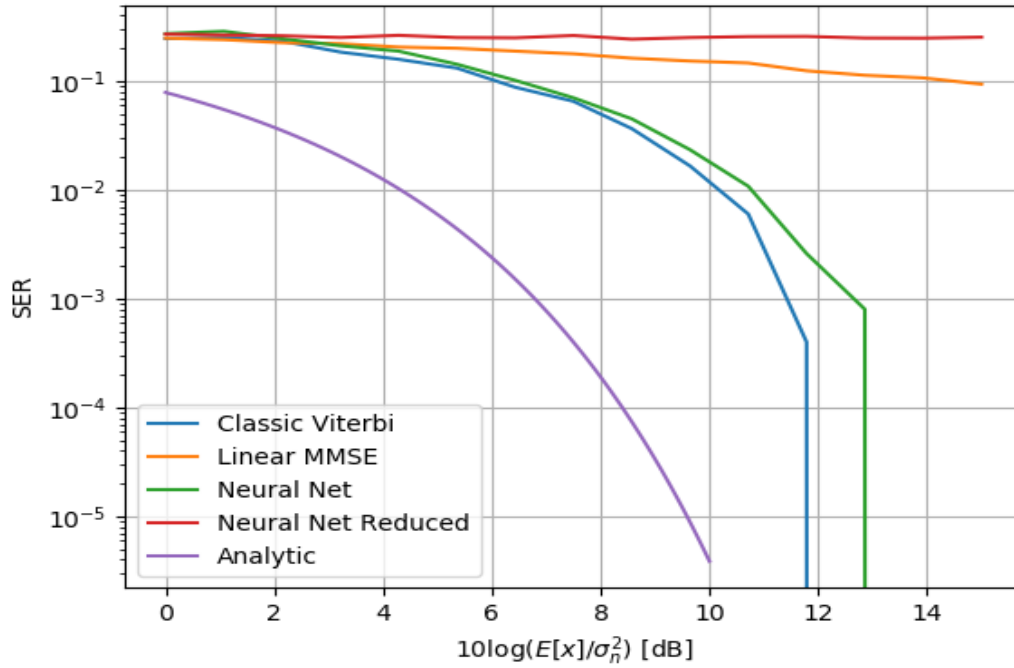

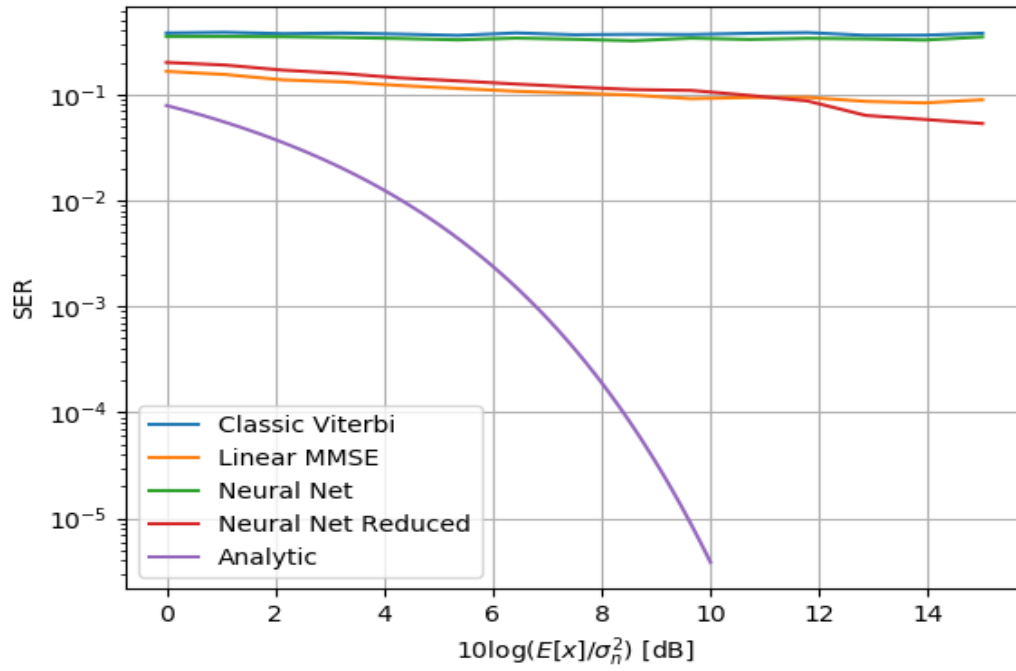Figure 1.6: Detection performance over LTI channel with AWGN

Figure 1.7: Detection performance over LTI channel with AWGN

We next consider the same LTI channels above with the receiver quantizer defined in equation (1.2) and plotted in 1.2.
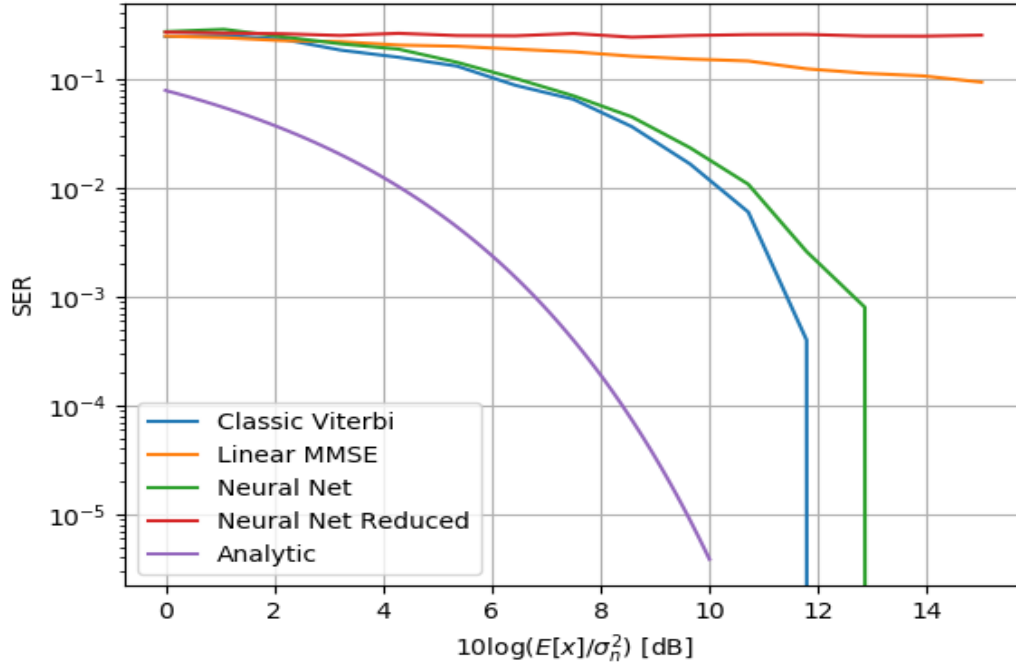
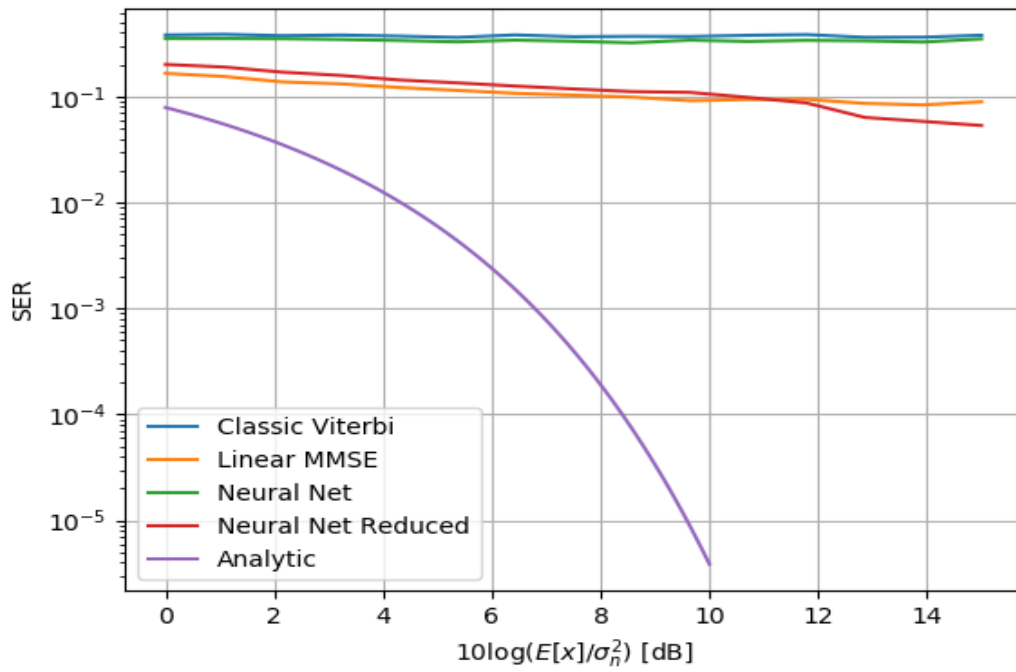Figure 1.8: Detection performance over quantized LTI channel with AWGN

Figure 1.9: Detection performance over quantized LTI channel with AWGN

1.5 Conclusion

In this work, a framework is developed to define the class of channels for channel estimation based on the previous work of [**?**] can be applied. The methods used in [**?**] are then extended in order to learn channels with addition types of randomness as well as offer potential complexity reductions in the resulting Viterbi Algorithm. In the simulation work, the results of [**?**] are extended to a non-linear channel. The proposed methods for complexity reduction are also simulated.

In future work, applying this general framework to related algorithms should be explored. In particular, the methods of [**?**] which were extended to include other communication detection

is evaluated in channels for which $p(y_i | \mathbf{x}_{i-L+1}^i)$, is unknown and non-linear. Simulation results indicate that this detector can successfully learn $p(y_i | \mathbf{x}_{i-L+1}^i)$ for non-linear channels. A method for reducing the complexity of the resulting Viterbi Algorithm is also proposed, however the method of state reduction for the channel needs to be further investigated to improve the poor performance observed.

In this work the MLSE is found using the Viterbi Algorithm. The same metrics can also be used in optimizing other performance metrics. In [**?**] these estimate metrics have been used for minimize bit error rate estimation using the BCJR algorithm.

TODO Extensions of specific work: More on reduced state. Using other channels and repeat online type of updates for dynamic channels for these more challenging channels. Learning channel changes in cases for which coherence time is shorter than updates to network made by pilot symbols.