Neural Network Based Decoding over Molecular Communication Channels

peter Hartig

February 5, 2020

Abstract

Contents

0.1 Notation

The following notation is used throughout this 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 random variable x. $\underset{x}{\text{argmin}} f(x)$ is the value of variable x which minimizes $f(x)$. Vectors are denoted by bold font $\mathbf{x}$ and are assumed to be column vectors unless noted. The vector $\mathbf{x}_{\text{i}}^{\text{j}}$ denotes a vector containing elements $\text{i} \rightarrow \text{j}$ of the original vector $\mathbf{x}$. $|\mathcal{A}|$ is used to denote the cardinality of a set $\mathcal{A}$.

0.2 Introduction

Characterizing and obtaining information about communication channels is a fundamental barrier to communication. While optimal and sub-optimal strategies for overcoming this barrier in many contexts have enabled vast and effective communication infrastructure, this barrier still limits communication in others. Molecular Communication channels pose a particularly difficult context in which to overcome this barrier as channel characteristics are often non-linear and may be dependent on the specific transmitted information. In communication contexts, such as wireless, "pilot" symbol-streams are often used to mitigate the difficultly in obtaining channel information by provide real-time information supporting an underlying channel model. The low symbol rate of Molecular Communication channels often makes such strategies impractical. However, the success of this data-driven technique in wireless channels suggest that perhaps an alternative, data-driven method may be viable in the Molecular Communication context. One potential data-driven method for characterizing these channels is a neural network. Neural networks have shown to be an effective tool in data-driven approximating of probability distributions.

The general communication channel is equivalent to a conditional probability $p(x|y)$, in which x is transmitted information and y is received information. $p(x|y)$ takes into account the (potentially random) channel through which the information x passes, and random noise added prior to receiving y. The communication problem entails optimizing a form of $p(x|y)$ over a set of possible, transmitted information x. In general, sub-optimal solutions do not require perfect knowledge of the distribution $p(x|y)$ and may be used when $p(x|y)$ is unknown or impractical to obtain. In this work, a neural network is used to estimate $p(x|y)$.

0.3 Background

0.3.1 MLSE

The form of $p(x|y)$ used as a detection objective function in this work

$$\underset{x \in \mathcal{A}}{\operatorname{argmin}}\, p(y|x) \tag{1}$$

is known as Maximum Likelihood Sequence Estimation (MLSE). This optimization over the set of all possible $x \in \mathcal{A}$ is exponentially complex in the cardinality of x, or as considered here, the number of the transmitted symbols. Additional information about the communication channel can, however, reduce this complexity. In order to illustrate, the following example is posed.

Consider the communication channel from which each received symbol in the sequence $\mathbf{y}$ is a causal, linear, and time invariant combination of a set of the transmitted symbol sequence $\mathbf{x}$ with coeffients $\mathbf{a}$.

$$y[k] = \sum_{\mathrm{l}=1}^{\mathrm{L}} a[l]x[k-l] \tag{2}$$

In this case, $p(\mathbf{y}|\mathbf{x})$ can be rewritten

$$\sum_{\mathrm{i}=1}^{\mathrm{N}} log(p(y_{\mathrm{i}}|\mathbf{x}_{\mathrm{i}-\mathrm{L}+1}^{\mathrm{i}})) \tag{3}$$

with independent components in the sum. In this case, problem (1) becomes simply finding the minimum path through a trellis (Figure). MLSE can thus be performed using the following algorithm over the trellis.

Viterbi Algorithm:

---

given $p(y_{\mathrm{i}}|x_{\mathrm{i}-\mathrm{L}+1}^{\mathrm{i}})\ \forall i \in 1..N$.
for $i = 1..N$
  for each state s **at time** i
    1. Let *survivor cost*$_s$ $+=$ min{incoming transition costs}
return Symbols corresponding to path of $\underset{s}{\operatorname{argmin}}$ *survivor cost*$_s$

For finite state, causal channels, MLSE reduces to the Viterbi Algorithm. Note that the Viterbi algorithm is *exponentially* complex in the number of channel states, but *linearly* complex in the length N of the transmitted sequence $\mathbf{x}$.

0.3.2 ViterbiNet

Despite the reduction in complexity offered by the Viterbi Algorithm for MLSE, the individual metrics used in each step of the algorithm $p(y_i|\mathbf{x}^i_{i-L+1})$ require knowledge of the channel which may be difficult to obtain. To estimate this distribution using a neural network, Baye's Rule is used.

$$p(y_i|\mathbf{x}^i_{i-L+1}) = \frac{p(\mathbf{x}^i_{i-L+1}|y_i)p(y_i)}{p(\mathbf{x}^i_{i-L+1})} \tag{4}$$

These terms can be interpreted as:

- $p(\mathbf{x}^i_{i-L+1}|y_i)$: The probability of being in channel state $\mathbf{x}^i_{i-L+1}$ given the corresponding received symbol from that time point y_i. If the number of states is finite, such a probability can be estimated using a neural network for classification of received signals into channel states.

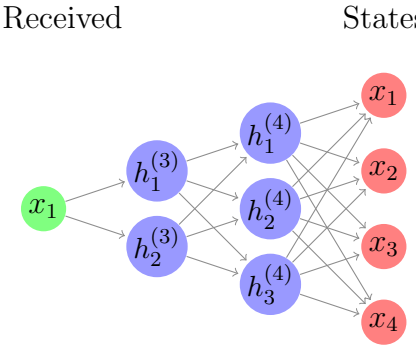


Figure 1: State classification Neural Network

- $p(y_i)$: The probability of the channel being in a given state (including any randomness in the channel state) as well as the probability of the noise. This probability distribution function can be estimated using a mixture-model (Fig. 0.3.2) based on a set of received signal training data (In this case the same data used to train the neural network). In particular, the Expectation Maximization Algorithm is used with a chosen model for the channel. This term is constant over all states in a given step of the Viterbi algorithm and can therefore be interpretted as a term indicating the importance of the given time point on the total sequence path cost.

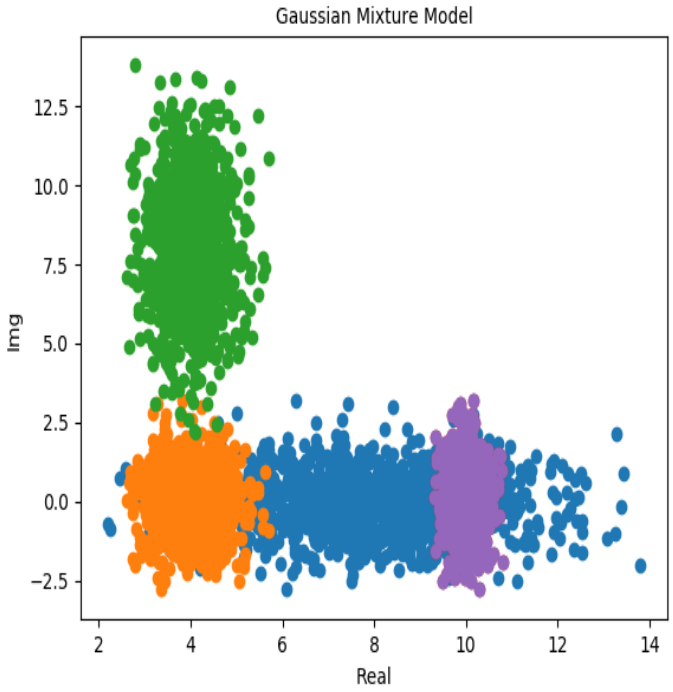


Figure 2: Mixture Model

- $p(\mathbf{x}^{\mathrm{i}}_{\mathrm{i-L+1}})$: The probability of a given transmitted sequence. This term can be neglected in the case of equiprobable symbols.

In summary, the metrics required by the Viterbi algorithm permit use of a neural network and mixture model to estimate the conditional probability distribution function $p(y_{\mathrm{i}}|\mathbf{x}^{\mathrm{i}}_{\mathrm{i-L+1}})$

Expectation Maximization Algorithm: (TODO Discuss how much derivation is needed here)

---

given $p(y_{\mathrm{i}}|x^{\mathrm{i}}_{\mathrm{i-L+1}})\ \forall i \in 1..N$.
for $i = 1..N$
    for each state s **at time** i
        1. Let *survivor* $cost_s$ $+=$ min{incoming transition costs}
return Symbols corresponding to path of $\underset{s}{\operatorname{argmin}}$ *survivor* $cost_s$

0.3.3 A Reduced State ViterbiNet

While the Viterbi Algorithm complexity complexity scales exponentially in the number of states possible at each step of the algorithm, in some cases this complexity can be reduced without significant degradation of performance. In particular, if the system is such that some states are redundant, these can be combined. The following relevant example with state redundancy is posed and one potential method of reducing these states is considered.

A received signal

$$y[k] = \sum_{l=1}^{L} a[l]x[k-l] + n[k],\ n[k] \sim \mathcal{N}(0,1) \tag{5}$$

with $x[k-l] \in \{-1, +1\}$ and $n[k] \sim \mathcal{N}(0,1)$.

In the case of a causal, LTI system with inter-symbol-interference $\mathbf{a} = [1, 0, .2, .2, .4]$ (with $\|\mathbf{a}\|_2^2 = 1$) the received signal (Fig.0.3.3) has fewer than the potential 2^5 states. As one of the channel taps is 0, this will have no impact and thus 16 of the potential 32 are removed. Further, as there are two taps with power 0.2 these can only represent 3 states.

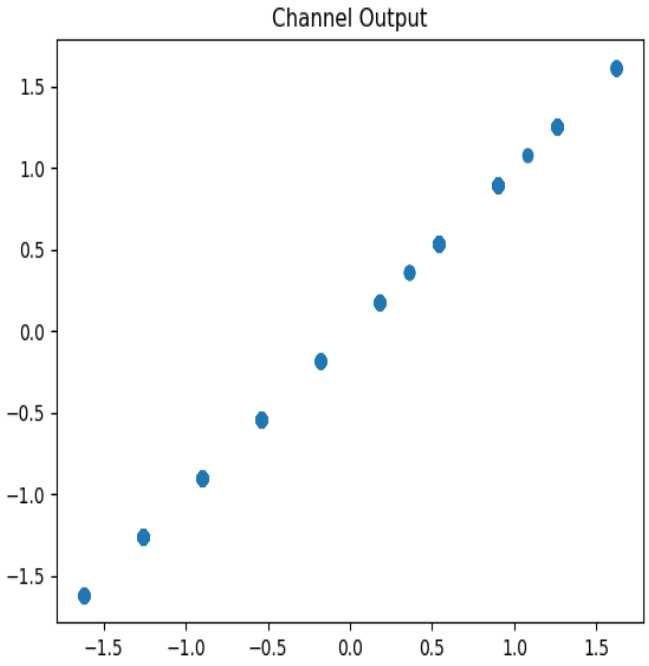


Figure 3: LTI channel with state redundancy with high SNR

One way to exploit state redundancy is to cluster the original 2^L states into a set of k clusters. The K-Means algorithm is proposed for clustering.

K-Means Clustering: A method for unsupervised classification

---

for Number of Iterations
  for each training point x_i, $\mathrm{i} \in \{1..N\}$
    1. Label x_i with index of closest centroid using $\|x_\mathrm{i} - \mathrm{closest}\|_2^2$
  for centroid L_c, $c \in \{1..C\}$
    1. Move L_c to average of all training points labeled "c"
return Centroid locations

Given a set of labeled training data, clustering reduces the number of channel states (and thus the Viterbi Algorithm Complexity) from $|\mathcal{A}|$ down to a number of clusters C. Choosing an appropriate number of clusters will influence the performance of the resulting decoder (Have figure showing this). A implementation point to note is that after clustering the training data, the number of states must be increased by $|\mathcal{A}|$. The Viterbi algorithm selects a symbol sequence corresponding to a path of state transitions through the trellis. In order to make the correspondence between a unique path and a symbol sequence, each state transition must represent a transmitted symbol. *Each* resulting centroid from the k-means clustering is associated to *each* potential transmit symbol in order to create a "shorter" trellis for which each state transition still represents a transmitted symbol. Cite state reduction paper as motivation.

Discuss minimum phase representations for channels and why this might be an advantage.

0.4 Simulation Results

0.4.1 System Model

Consider the received signal

$$y[k] = \sum_{\mathrm{l}=1}^{\mathrm{L}} a[l]x[k-l] + n[k],\ n[k] \sim \mathcal{N}(0,1) \tag{6}$$

with $x[k-l] \in \{-1,+1\}$ and $n[k] \sim \mathcal{N}(0,1)$. The resulting signal to noise ratio (SNR) is $\frac{E\{x[k]\}}{\sigma^2}$.

Unless noted, the neural network architecture and training are characterized by:

- Architecture: 4 layers 1, 100, 50, M^L
- Training Data Examples: 5000
- Dropout Rate: .5
- Neural Network Updates: Adam with step size 10^{-3} [**?**]
- Batch Size: 30
- Backpropogation Updates (Epochs): 300
- Loss Function: Cross Entropy

Figure 4: Simulation Channels: LTI Channel

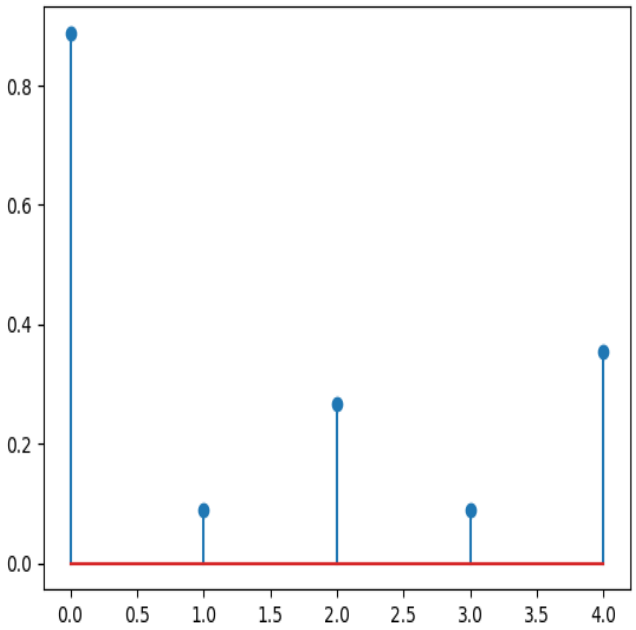
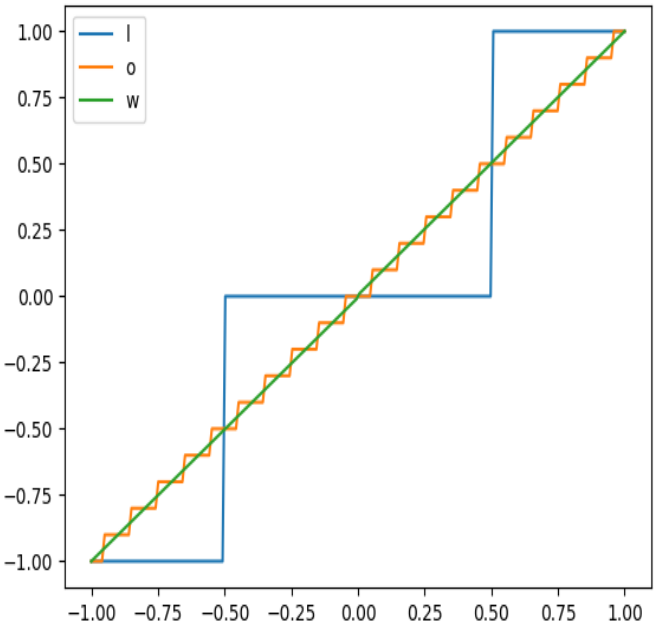


Figure 5: Simulation Channels: Quantizer



Adding quantization at matched filter (prior to noise being added)
Details of NN architecture and training
Details of Mixture Model training
Consider the received signal

$$y[k] = \sum_{l=1}^{L} a[l]x[k-l] + n[k],\ n[k] \sim \mathcal{N}(0,1) \tag{7}$$

with $x[k-l] \in \{-1,+1\}$ and $n[k] \sim \mathcal{N}(0,1)$. The resulting signal to noise ratio (SNR) is $\frac{E\{x[k]\}}{\sigma^2}$.

Figure 6: Simulation Channels: LTI Channel

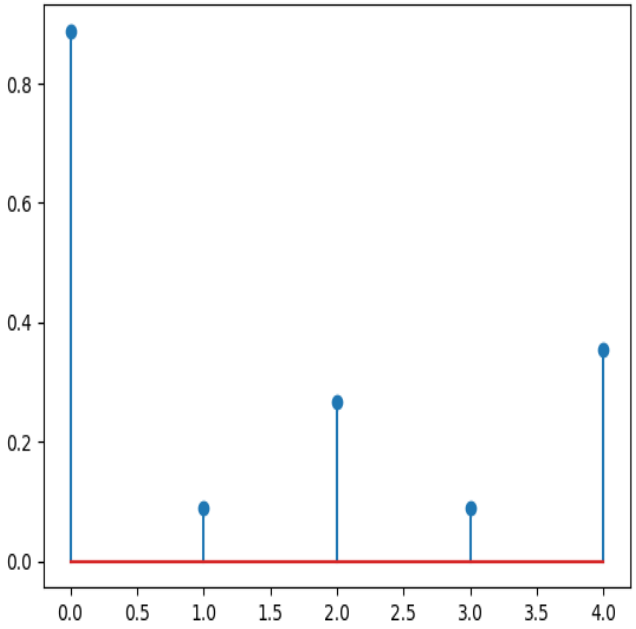
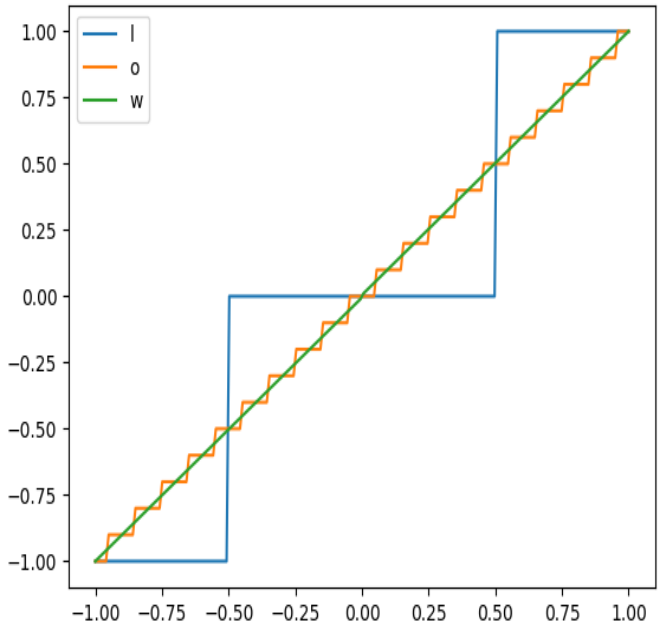


Figure 7: Simulation Channels: Quantizer



Adding quantization at matched filter (prior to noise being added)
Details of NN architecture and training
Details of Mixture Model training

0.4.2 Results

proposed Figures

Figure 8: LTI Channel performance

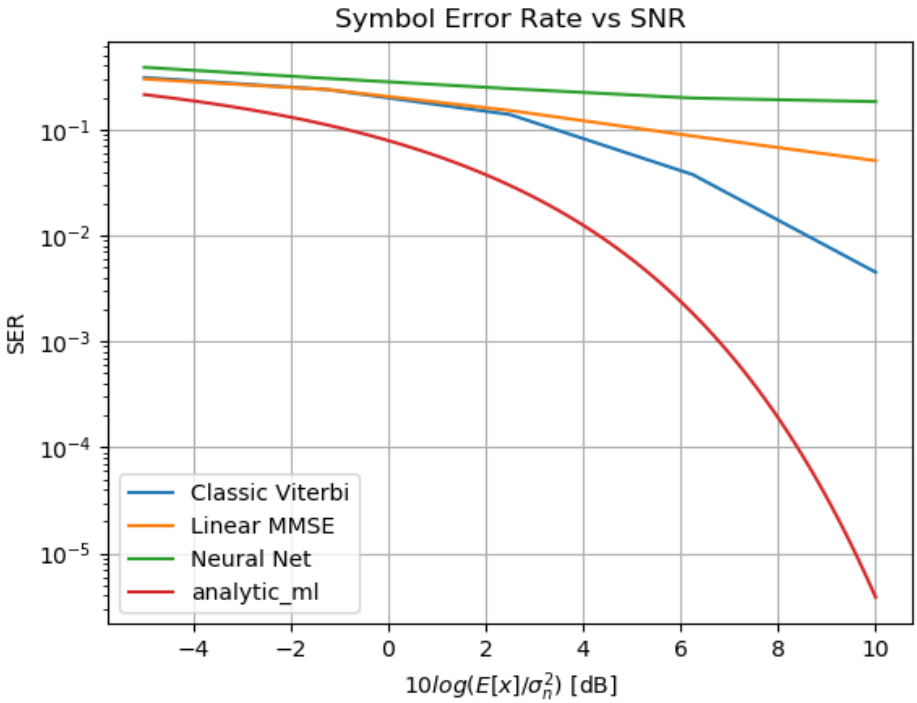


Figure 9: LTI + Quantized Channel performance

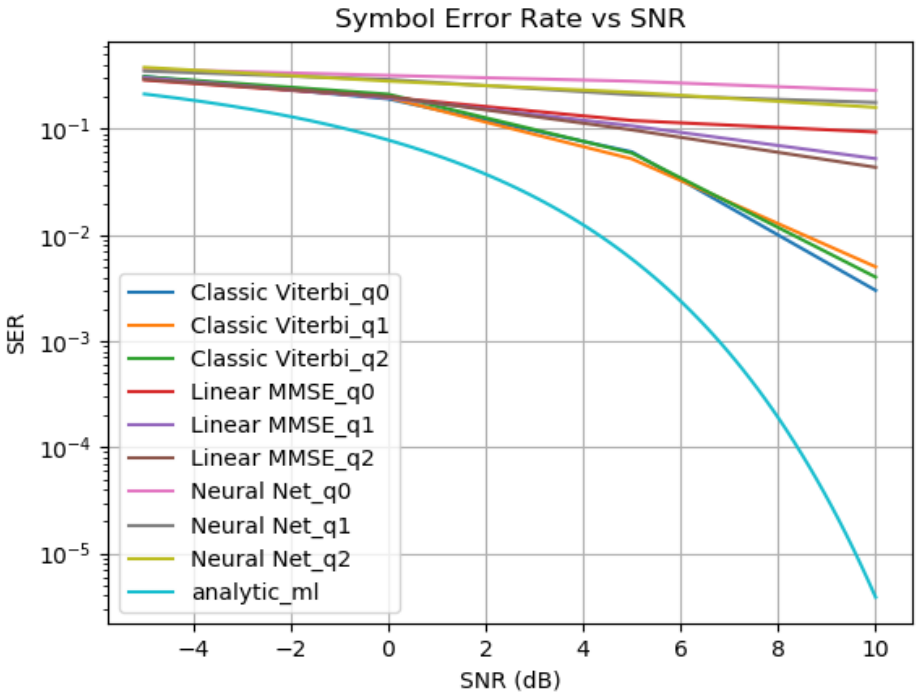
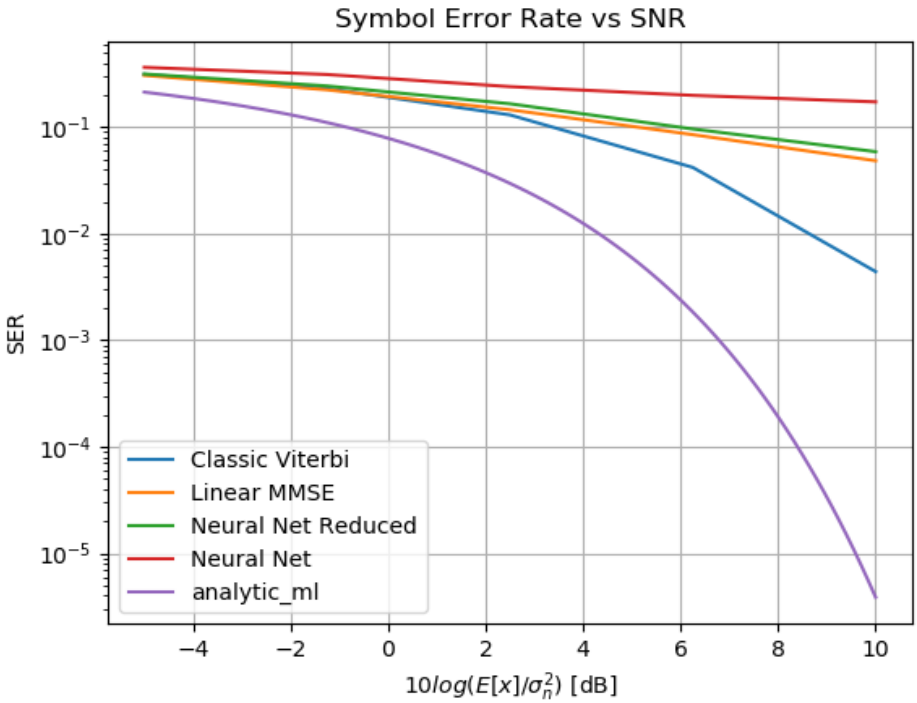
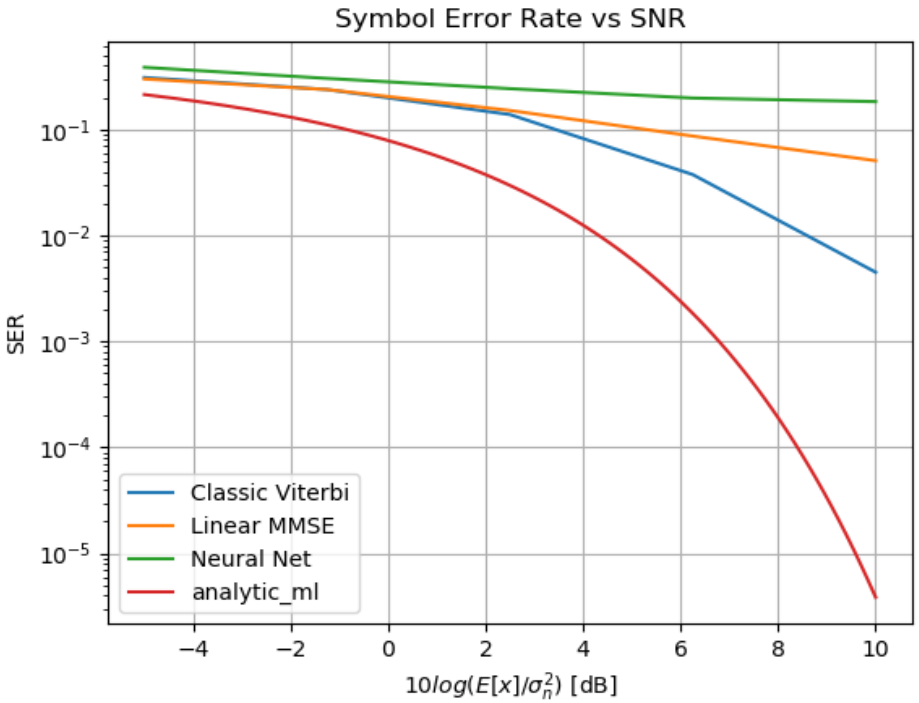


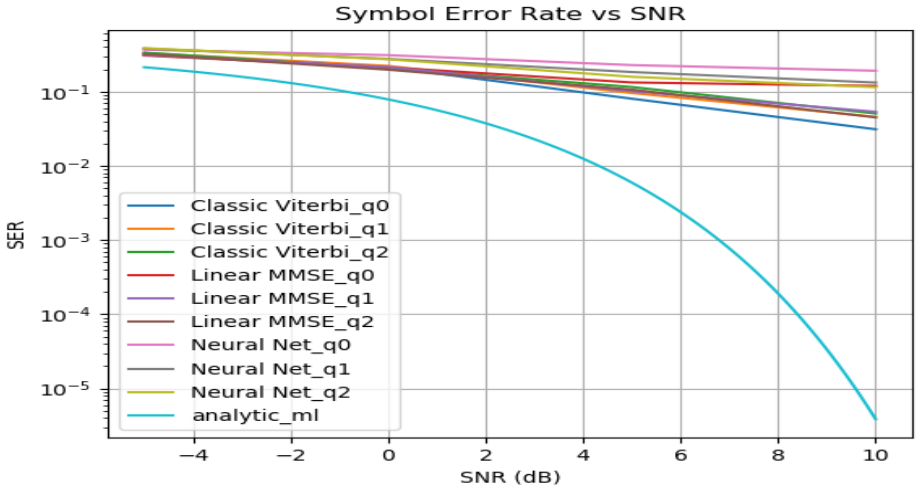
Figure 10: Reduced State ViterbiNet: LTI Channel



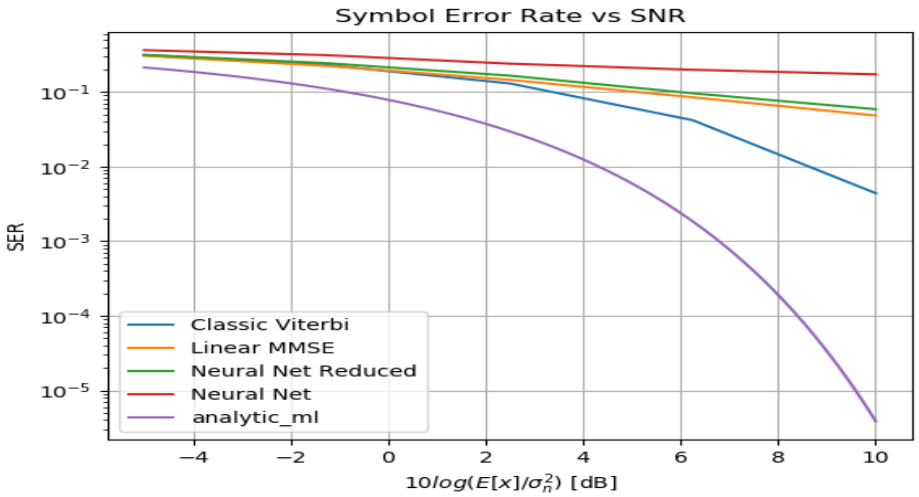
- ViterbiNet performance compared to MMSE and classic Viterbi LTI Channel



- ViterbiNet performance compared to MMSE and classic Viterbi non-linear Channel



- Reduced ViterbiNet on LTI Channel



- Reduced ViterbiNet on non-linear Channel

ViterbiNet

Reduced State ViterbiNet

0.5 Conclusion

0.5.1 Future Work

Discuss forward backward (App) algorithms that could be implemented.