Neural Network Based Decoding over Molecular Communication Channels

Peter Hartig

February 12, 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 a 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. The vector $\mathbf{x}_{\mathrm{i}}^{\mathrm{j}}$ denotes a vector containing the subset $\mathrm{i} \rightarrow \mathrm{j}$ of the original vector $\mathbf{x}$. $|\mathcal{A}|$ denotes the cardinality of the set $\mathcal{A}$.

0.2 Introduction

Characterizing and obtaining information about the probability distribution governing communication channels is a fundamental barrier to communication. While optimal and sub-optimal strategies for overcoming this barrier have enabled vast and effective communication infrastructure, this barrier still limits communication in other contexts. One such context is the Molecular Communication channel which may be non-linear, time variant, and dependent on the specific transmitted information. Communication contexts, such as wireless, use "pilot" symbol-streams to mitigate the difficultly in obtaining channel information by providing 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 successful for the Molecular Communication channel. 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 $p(x|y)$, with transmitted information x and received information y. $p(x|y)$ 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 form of the optimization $\underset{x \in \mathcal{A}}{\text{argmin}} f(x)$, in which $\mathcal{A}$ is the set of possible 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 supported estimate of $p(x|y)$ is investigated.

0.3 Background

0.3.1 MLSE and the Viterbi Algoithm

In this work, we use the form of $p(x|y)$ below as a detection objective function. The optimization problem

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

is also known as Maximum Likelihood Sequence Estimation (MLSE). The size of the search space $|\mathcal{A}_1^{\mathrm{N}}|$ is grows exponentially in the length of x, 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 as

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

Noting that terms in the sum are independent, problem (1) becomes simply finding the shortest path through a trellis with edges weighted with $log(p(y_{\mathrm{i}}|\mathbf{x}_{\mathrm{i-L+1}}^{\mathrm{i}}))$ (Figure of trellis). For finite state, causal channels, MLSE reduces to the Viterbi Algorithm. (TODO add figure here).

Viterbi Algorithm:

---

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

Note that the Viterbi algorithm is *exponentially* complex in the number of channel states $|\mathcal{A}_{\mathrm{i-L+1}}^{\mathrm{i}}|$, but *linearly* complex in the length 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 $p(y_i|\mathbf{x}_{i-L+1}^i)$ require knowledge of the channel probability distribution which may be difficult to obtain. To estimate this distribution using a neural network, Baye's Rule is used.

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

- $p(\mathbf{x}_{i-L+1}^i|y_i)$: The probability of being in channel state $\mathbf{x}_{i-L+1}^i$ given the corresponding received symbol y_i. If the number of states is finite, such a probability can be estimated using a neural network for classification. The output of such a network is a probability distribution over the possible states of the channel for a received symbol.

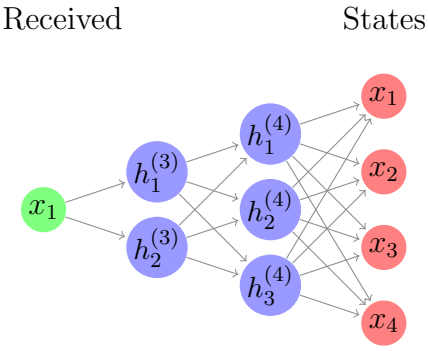


Figure 1: State classification Neural Network

- $p(y_i) = \sum_{s\in\mathcal{S}} p(s, y_i)$: The joint probability of channel state s and the received signal y_i marginalized over all channel states $\mathcal{S}$. For the case of the LTI channel above, each state s corresponds to a possible symbol sequence $\mathbf{x}_{i-L+1}^i$. This probability distribution function can be estimated using a mixture-model (Fig. 0.3.2) based on training data of received signals (In this case the same data used to train the neural network). In particular, the Expectation Maximization Algorithm (CITE) is used with a chosen model for the channel states and noise.

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

---

given $p(y_i|x_{i-L+1}^i)\ \forall i \in 1..N$.
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 $\underset{s}{\text{argmin}}$ *survivor cost*$_s$

As $p(y_i)$ is constant over all states in a given step of the Viterbi algorithm, this may be interpreted as a weighting factor for the cost of one received symbol in the total path cost. (More detail on this or enough?)

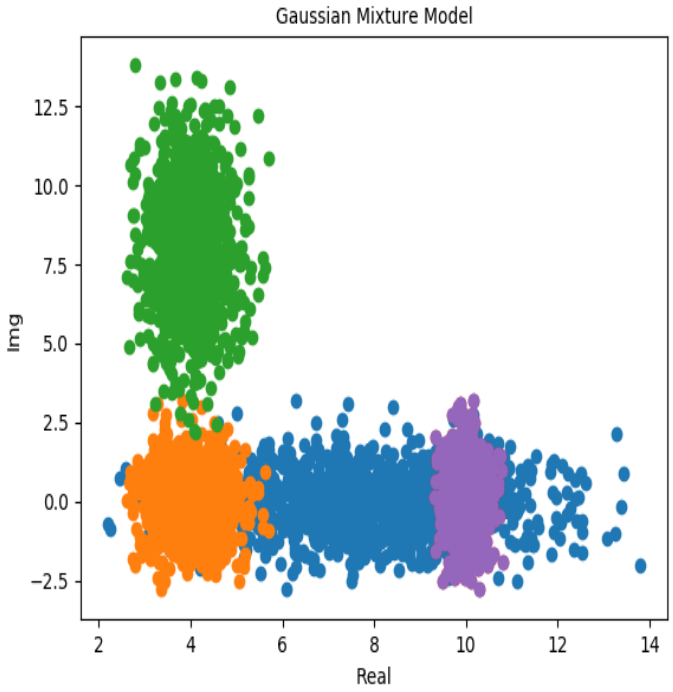


Figure 2: Mixture Model

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

In summary, the metrics $p(y_i|\mathbf{x}_{i-L+1}^i)$ required by the Viterbi algorithm can be estimated using a neural network and a mixture model.

0.3.3 A Reduced State ViterbiNet

While the Viterbi Algorithm complexity 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 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 the number of states is considered.

The received signal is

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

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 characterized by $\mathbf{a} = [a[1]...a[5]] = [1,0,.2,.2,.4]$ ($\|\mathbf{a}\|_2^2 = 1$), the received signal (Fig.0.3.3) has fewer than the potential $|\mathcal{A}^{\mathrm{i}}_{\mathrm{i-L+1}}| = 2^5$ 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 of value 0.2, these can represent only 3 states rather than the expected 4 states.

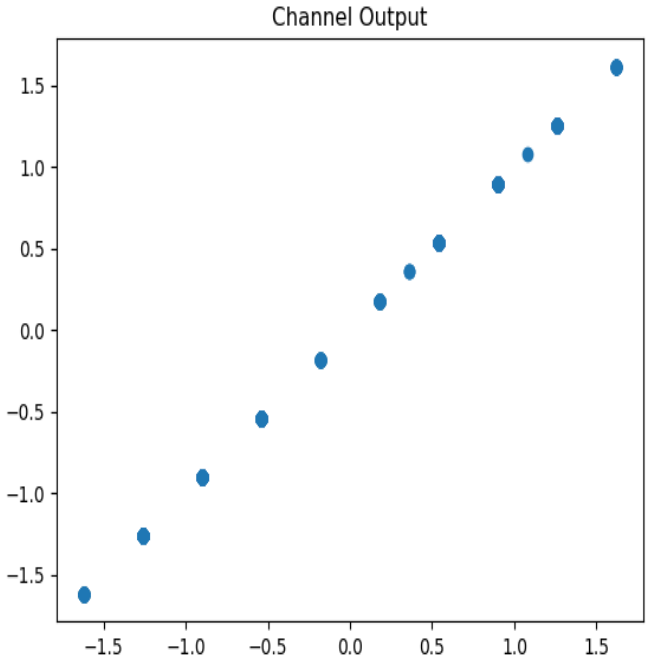


Figure 3: LTI channel with state redundancy with high SNR

One way to exploit state redundancy is to cluster the original states $|\mathcal{A}_{\mathrm{i-L+1}}^{\mathrm{i}}|$ states into a set of k clusters using training data. The K-Means algorithm is proposed as one such method.

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

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 $|\mathcal{A}_{\mathrm{i}}|$. 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.

0.3.4 The Molecular Communication Channel

Provide necessary background for any nano-particle data shown in results.

0.4 Results

0.4.1 A supervised equalization benchmark

To provide a linear equalization reference to the non-linearity of the proposed neural network, mixture model equalization, the Linear, Minimum, Mean-Squared Error (LMMSE) equalizer is used in the results. The convex and continuously differentiable optimization problem

$$\underset{\mathbf{h}\in\mathcal{C}_{lengthx1}}{\operatorname{argmin}} \; E[\|x-\mathbf{h}^T\mathbf{y}\|^2],$$

can be solved by

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

to give the optimal LMMSE equalizer

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

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

$$E[\mathbf{R}_{yy}] = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathbf{y}\mathbf{y}^{\mathrm{H}} and$$

$$E[\mathbf{r}_{yx}] = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathbf{y}x;$$

using the same training data as the neural network and mixture model. Note that just as with the neural network, making this estimate requires knowledge (or estimation) of the channel memory length l.

0.4.2 Simulation Details

System Model

Verification of the implementation and integration of the algorithms described above is based on the following system model.

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

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

Algorithm Parameters

The parameters for the various algorithms used in this work are detailed below

- **Neural Network**
  - Architecture: 4 layers 1, 100 (Tanh activation), 50 (relu activation), M^L (Softmax $\rightarrow$ Negative Log-Likelihood)
  - 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
- **Expectation Maximization Algorithm**
  - Training Data Examples: 5000
- **K-Means Algorithm (For Reduced State ViterbiNet)**
  - Training Data Examples: 5000

0.4.3 Simulation Results

Figure 4: Simulation Channels: LTI Channel

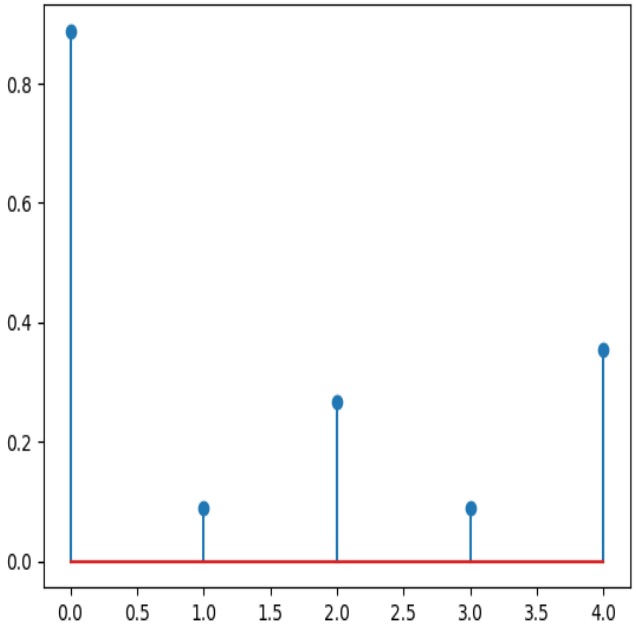
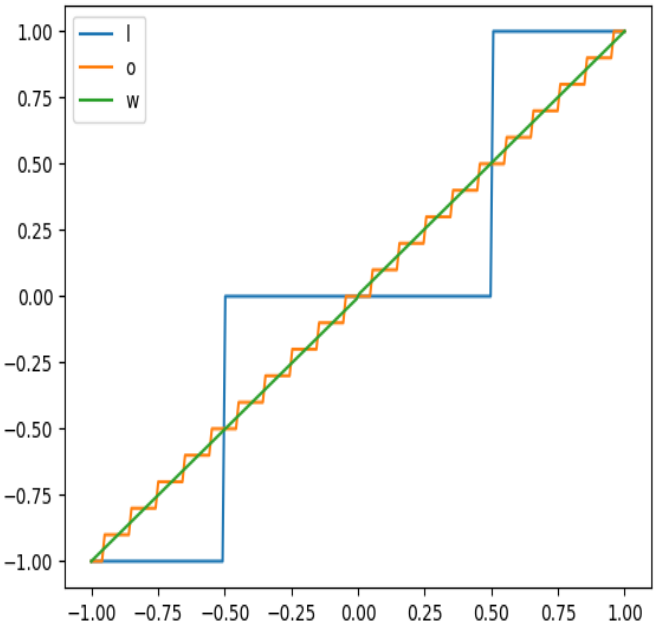


Figure 5: Simulation Channels: Quantizer



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

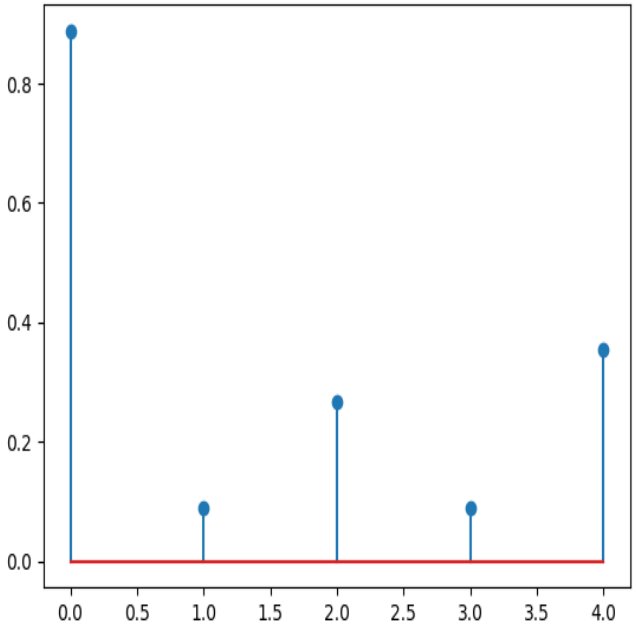
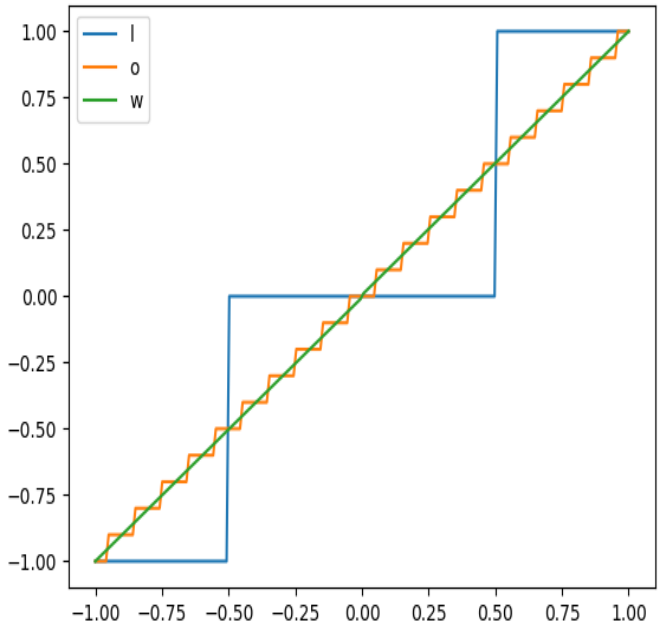


Figure 7: Simulation Channels: Quantizer



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

0.4.4 Results

proposed Figures

Figure 8: LTI Channel performance

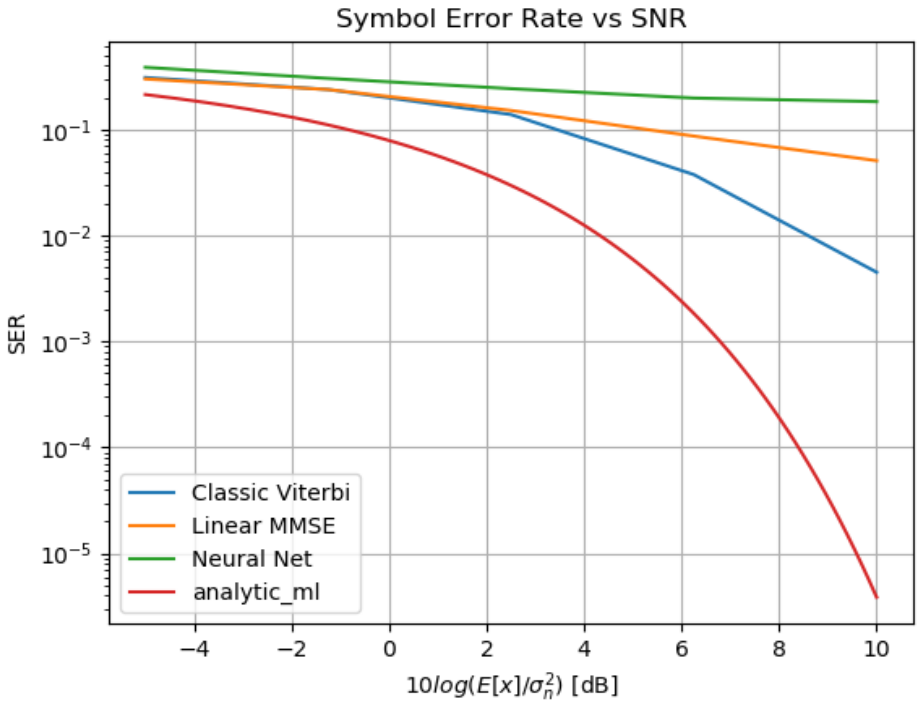


Figure 9: LTI + Quantized Channel performance

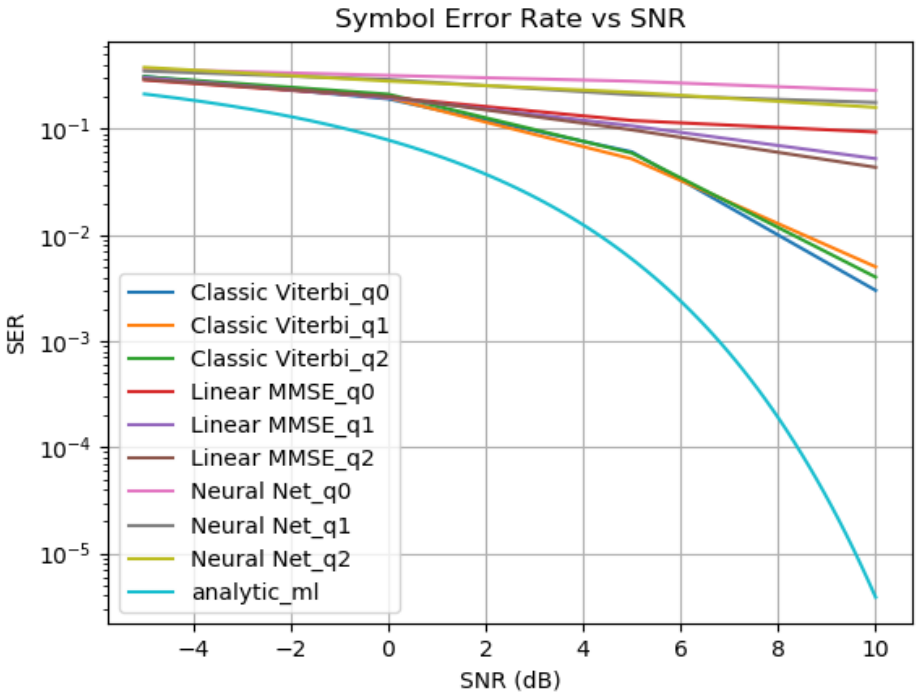
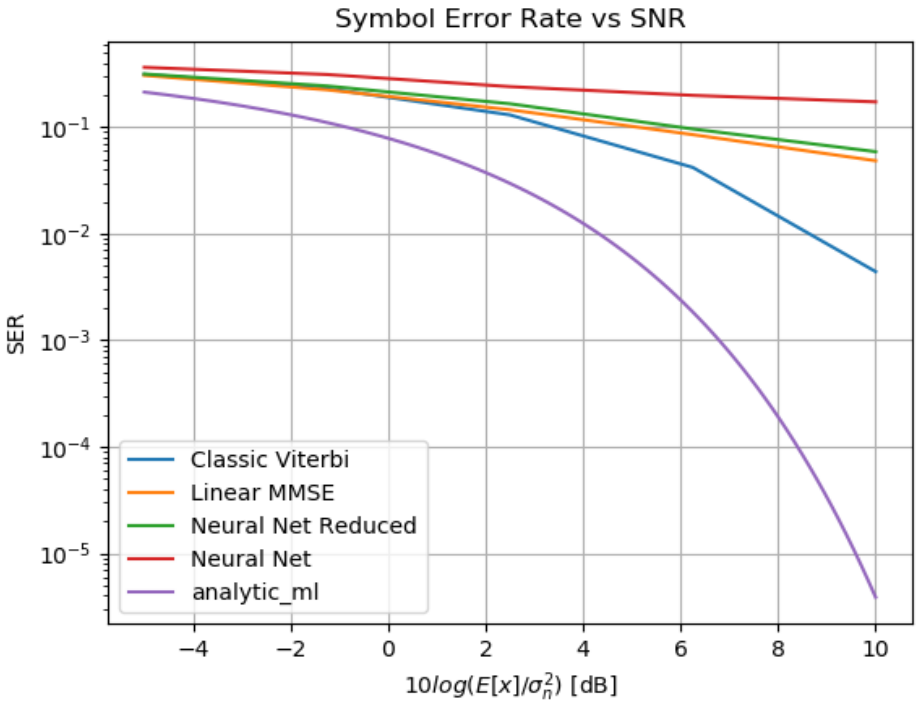
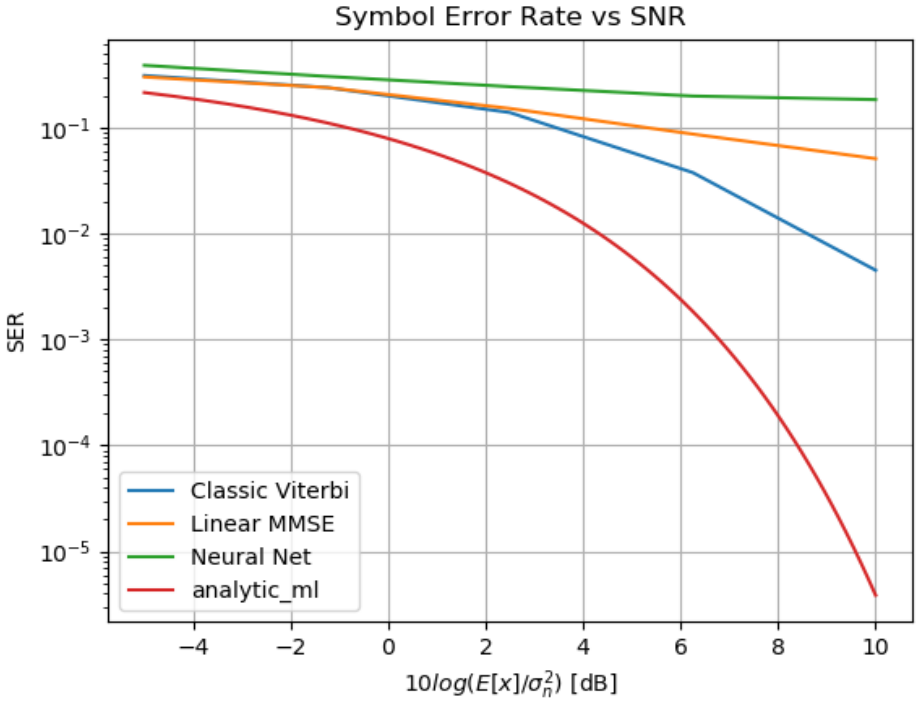


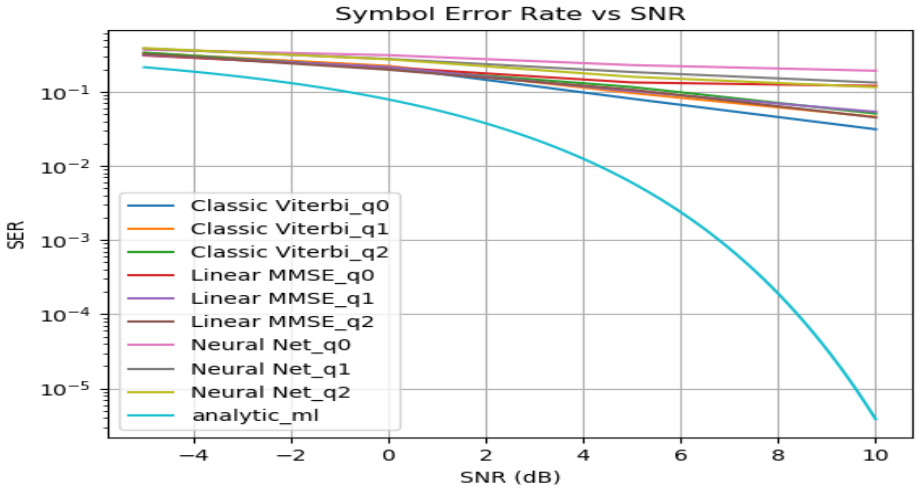
Figure 10: Reduced State ViterbiNet: LTI Channel



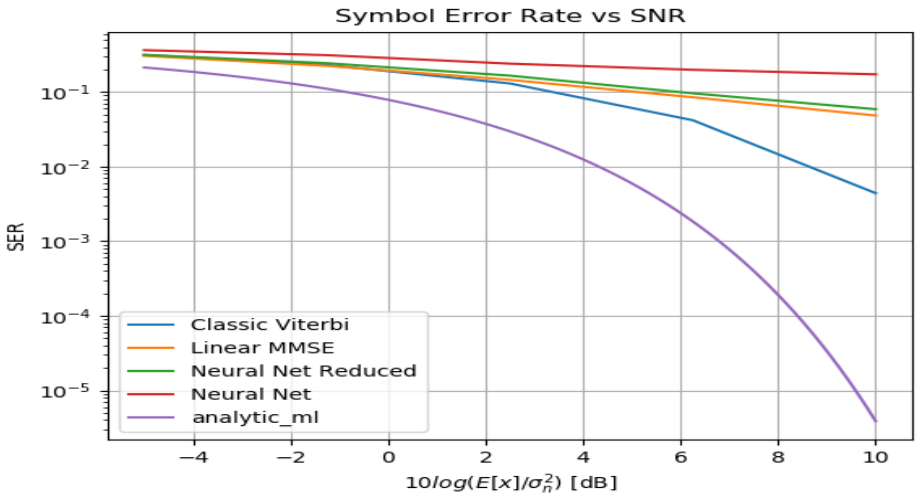
- ViterbiNet performance compared to MMSE and classic Viterbi LTI Channel



- 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.