Neural Network Based Decoding over Molecular Communication Channels

Peter Hartig

January 31, 2020

Abstract

Contents

0.1 Notation

The following notation conventions are used Expectation Conditional Probability Argmax Vector Indexing

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 for detection in this work is $\underset{x}{\operatorname{argmin}}\ P(y|x)$. This optimization, known as Maximum Likelihood Sequence Estimation (MLSE), over the set of all possible x is exponentially complex in the cardinaltiy of x . Information about the communication channel can, however, reduce the complexity of this problem. In order to illustrate this reduction, the following example is proposed.

Consider the communication channel over which a causal, linear, and time invariant combination of a set of the transmitted information is received.

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

In this case, $P(y|x)$ can be rewritten as $\prod P(y_{\mathrm{i}}|x^{\mathrm{i}}_{\mathrm{i-L+1}}) = \sum log(P(y_{\mathrm{i}}|x^{\mathrm{i}}_{\mathrm{i-L+1}}))$. The sequence of received symbols $\mathbf{y}$ can be equivalently represented the by trellis:

TODO IMPORT Trellis picture in which each time-point k represents a unique set of L transmitted symbols $x[k-l]\ \forall l \in 1..L$. Due to the finite number of states that channel can be in at a given time, MLSE can be performed using the Viterbi Algorithm over this trellis.

Viterbi Algorithm:

```
given $P(y_{\mathrm{i}}|x^{\mathrm{i}}_{\mathrm{i-L+1}})\ \forall i \in 1..N$ .
Let $\lambda := \lambda^{k-1}$.
for $i = 1..N$
    for each state $s$ at time $i$
        1. Let Path$_s := \mathbf{prox}_{\lambda g}(x^k - \lambda \nabla f(x^k))$.
        2. break if $f(z) \le \hat{f}_\lambda(z, x^k)$.
        3. Update $\lambda := \beta\lambda$.
return $\underset{s}{\operatorname{argmin}}\ cost[i]$, $x^{k+1} := z$.
```

For finite state, causal channels, MLSE reduces to the Viterbi Algorithm. Note that while Viterbi algorithm does have exponential complexity in the length of the channle (i.e. L), complexity is linear in the length of the sequence.

0.3.2 ViterbiNet

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

$$P(y_\mathrm{i}|x_{1\ldots\mathrm{L}}) = \frac{P(x_{1\ldots\mathrm{L}}|y_\mathrm{i})P(y_\mathrm{i})}{P(x_{1\ldots\mathrm{L}})} \tag{2}$$

These terms can be interpreted as:

- $P(x_{1\ldots\mathrm{L}}|y_\mathrm{i})$: The probability of being in a channel state given the corresponding received symbol from that time point. In the case of a finite number of states, such a probability can be estimated using a neural network for classification of received signals into channel states.

- $P(y_\mathrm{i})$: The randomness of the channel. As each received signal represents a state of the system to which noise may be added. A representative mixture-model can be estimated using a set of received signal training data.

- $P(x_{1\ldots\mathrm{L}})$: Assuming the transmitted symbols are equiprobable, this term can be neglected as all states $x_{1\ldots\mathrm{L}}$ will have equal probability.

0.4 Simulation Results

0.4.1 System Model

Consider the received signal

$$y[k] = \sum_{\mathrm{l}=1}^{\mathrm{L}} a[l]x[k-l] + n[k],\ n[k] \sim \mathcal{N}(0,1) \tag{3}$$

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 1: Simulation Channels: LTI Channel

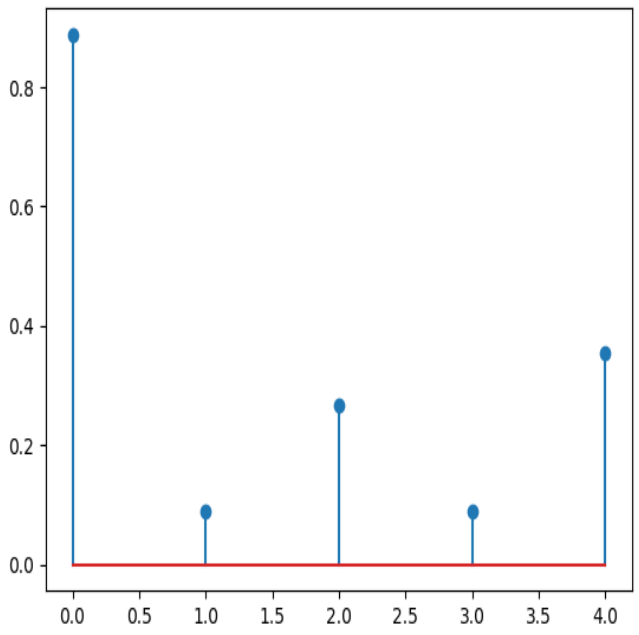
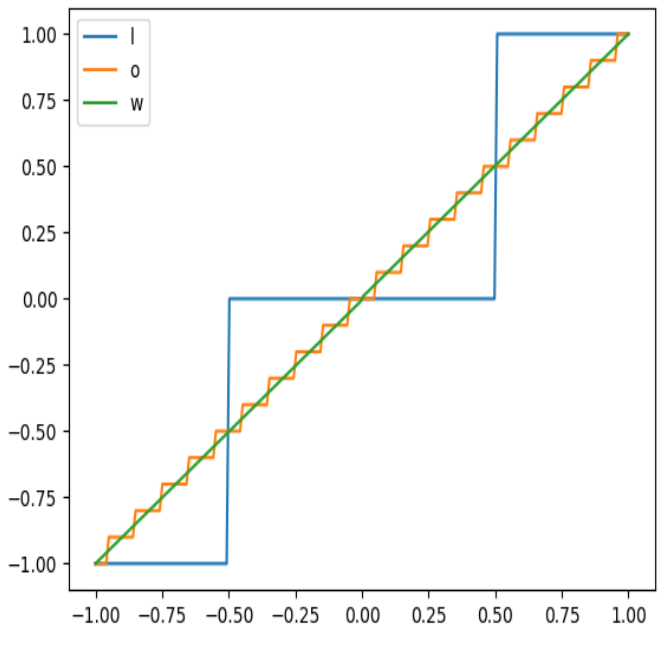


Figure 2: Simulation Channels: Quantizer



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 3: LTI Channel

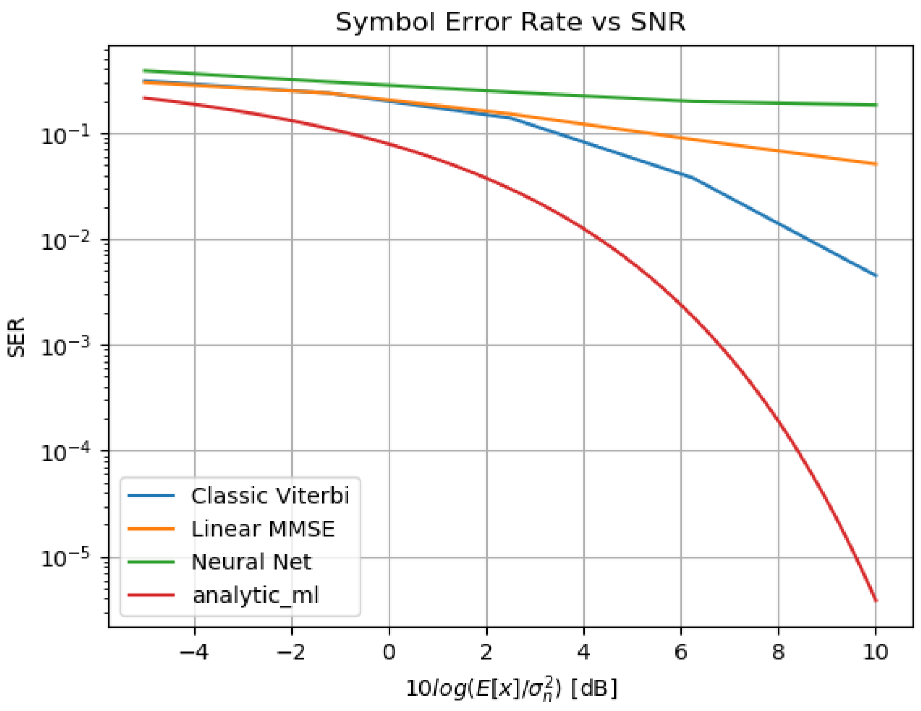


Figure 4: LTI + Quantized Channel

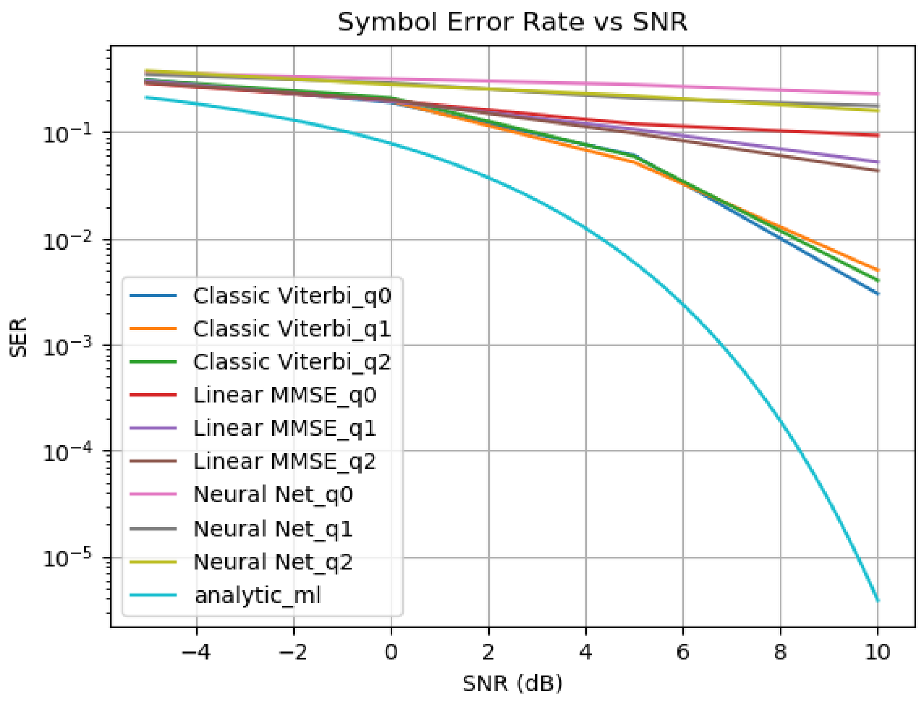


Figure 5: Reduced State (32 → 4) ViterbiNet: LTI Channel

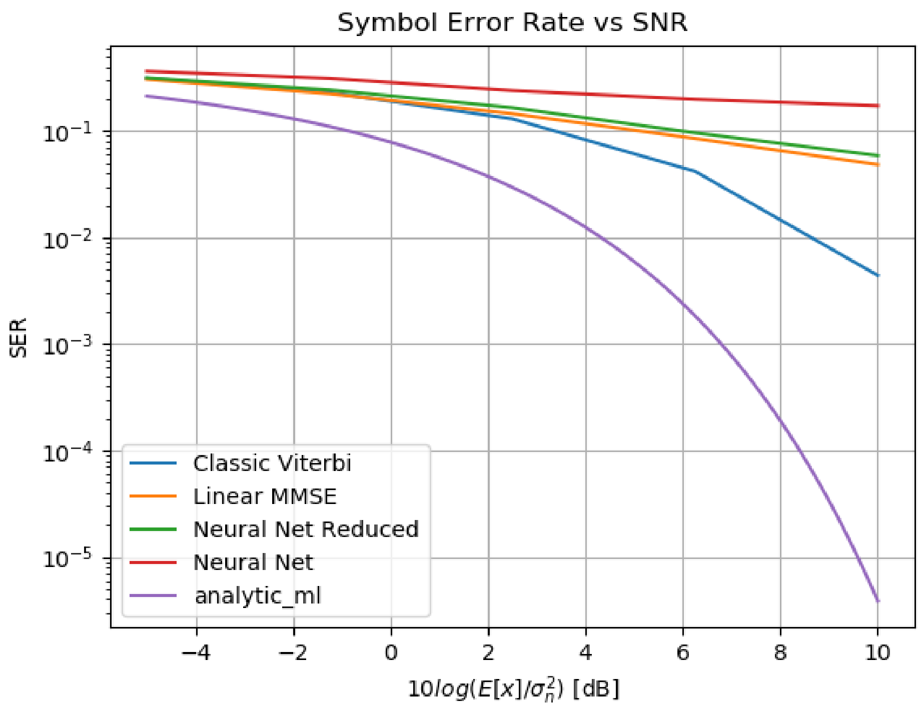
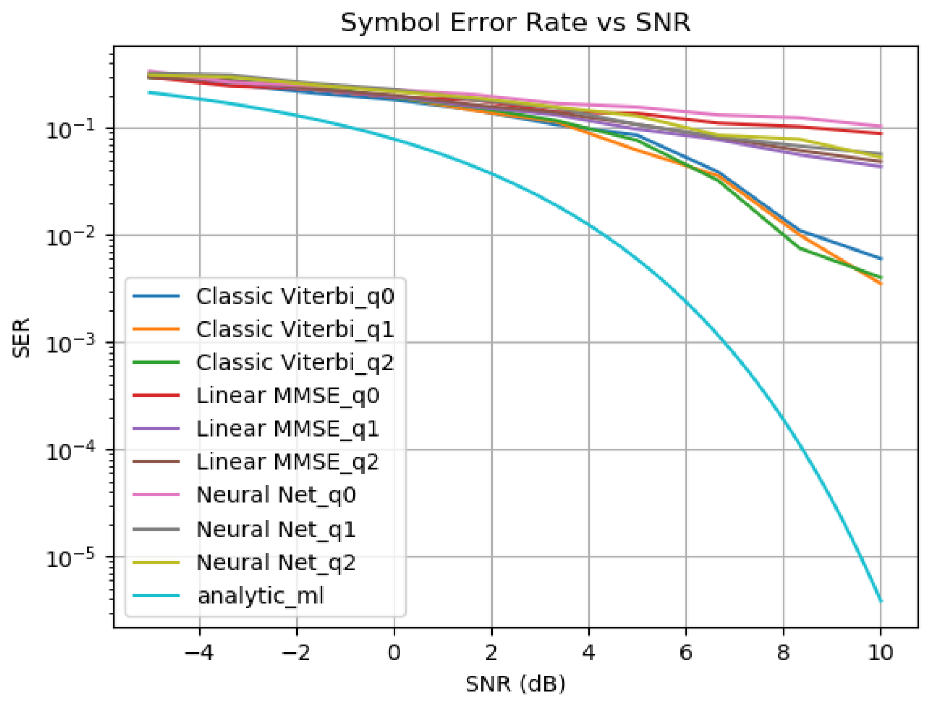


Figure 6: Reduced State (32 $\rightarrow$ 4) ViterbiNet: LTI + Quantized Channel



ViterbiNet

Reduced State ViterbiNet

0.5 Conclusion

0.5.1 Future Work

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