Neural Network Based Sequence Detection over Molecular Mommunication Channels

Peter Hartig

March 1, 2020

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.

Contents

0.1 Notation

This section details the notation is 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 . $x \sim \mathcal{N}(0,1)$ 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 $\mathbf{x}_{\mathrm{i}}^{\mathrm{j}}$ 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 $|\mathcal{A}|$. $\underset{x}{\operatorname{argmin}}\, f(x)$ is the value of variable x which minimizes the function $f(x)$.

0.2 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 $\mathbf{y}$ [1, 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 $p(\mathbf{x}|\mathbf{y})$ [9]. 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, 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 [7] [8]. 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 [3].

0.3 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 with a transmitter sending a sequence of information $\mathbf{x}$ over a channel. At the receiver, the sequence $\mathbf{y}$ is detected. The elements of $\mathbf{x}$ are chosen from a finite symbol alphabet $\mathcal{A}$ such that $x[\mathrm{k}] \in \mathcal{A}$.

Figure 1: The point to point communication system.

The components of the vector $\mathbf{y}$ are described by

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

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

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

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

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

In the first communication channel $f_{\mathrm{k}}(\mathbf{x}) + n[\mathrm{k}]$ considered, each $y[\mathrm{k}]$ is a causal, linear, and time-invariant (LTI) combination of the transmitted

sequence $[x[\mathrm{k}], x[\mathrm{k}-1]...x[\mathrm{K}-\mathrm{L}+1]]$ weighted by coefficients $[a[0], a[1]...[\mathrm{L}-1]]$.

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

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

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

with

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

for

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

A quantizer with 20 quantization levels and lower/upper saturation levels -5/+5 respectively is shown in Figure 7.

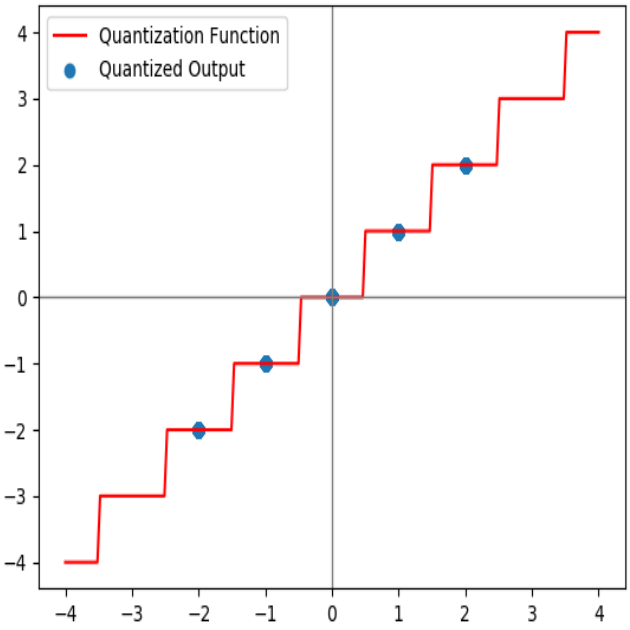

Figure 2: Quantizer with 20 quantization levels, lower saturation levels = -5 and upper saturation levels = $+5$

.

0.4 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 introduced. 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. Lastly, a linear equalizer is derived to be used as a reference in the simulation results.

0.4.1 MLSE and the Viterbi Algorithm

Rather than considering only the transmitted symbols $\mathbf{x}$ for the MAP problem described above, instead consider a system in which a sequence $\mathbf{y}$ is received from a channel which may take on a set of states $\{s_1, s_2 ... s_\mathrm{L}\} = \mathcal{S}$. With knowledge of the potential states $\{s_1, s_2 ... s_\mathrm{L}\}$ but not the sequence of

realized states $[s[1], s[2]...s[\mathrm{N}]]$ corresponding to $[y[1], y[2]...y[\mathrm{N}]]$, the MAP problem is restated as

$$\underset{\mathbf{s}\in\mathcal{S}^N}{\operatorname{argmax}}\, p(\mathbf{s}|\mathbf{y}).$$

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

$$\underset{\mathbf{s}\in\mathcal{S}^N}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{s})p(\mathbf{s}) \tag{1}$$

will be used in the remainder. Noting that the size of the search space $|\mathcal{S}^N|$ 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 $s_i \in \{s_1, s_2...s_{\mathrm{L}}\}$ represents a possible transmitted sequence $\mathbf{x}^{\mathrm{i}}_{\mathrm{i-L+1}}$ and $p(\mathbf{x}) = p(\mathbf{s})$. If all symbols in $\mathcal{A}$ are selected with equal probability, as is commonly assumed to maximize information entropy for finite $|\mathcal{A}|$, the term $p(\mathbf{x})$ can be removed from problem 1 without effect. For this channel,

$$\underset{\mathbf{s}\in\mathcal{S}^N}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{s})p(\mathbf{s}) = \underset{\mathbf{s}\in\mathcal{S}^N}{\operatorname{argmin}} \sum_{\mathrm{i}=1}^{\mathrm{N}} -\log(p(y[\mathrm{i}]|\mathbf{s}^{\mathrm{i}}_{\mathrm{i-L+1}})p(\mathbf{s}^{\mathrm{k+1}}_{\mathrm{l}}|\mathbf{s}^{\mathrm{k}}_{\mathrm{j}})) =$$

$$\underset{\mathbf{x}\in\mathcal{A}^N}{\operatorname{argmin}} \sum_{\mathrm{i}=1}^{\mathrm{N}} -\log(p(y[\mathrm{i}]|\mathbf{x}^{\mathrm{i}}_{\mathrm{i-L+1}})p(\mathbf{x}^{\mathrm{k+1}}_{\mathrm{l}}|\mathbf{x}^{\mathrm{k}}_{\mathrm{j}}))$$

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

$$-\log(p(y[\mathrm{k}+1]|\mathbf{s}^{\mathrm{k+1}}_{\mathrm{i}})p(\mathbf{s}^{\mathrm{k+1}}_{\mathrm{i}}|\mathbf{s}^{\mathrm{k}}_{\mathrm{j}})).$$

This is illustrated in the following example.

Consider the communication system with $\mathcal{A} = \{0, 1\}$ and received symbols

$$y[\mathrm{k}] = a[0]x[\mathrm{k}] + a[1]x[\mathrm{k}-1] + n[\mathrm{k}],$$

with $n[\mathrm{k}] \sim \mathcal{N}(0,1)$. The resulting state graph (or trellis) is depicted in Figure 3. Again assuming equiprobable $\mathbf{x}_{\mathrm{i-1}}^{\mathrm{i}} \in \mathcal{A}^2$, the value of $p(s_{\mathrm{j}}^{\mathrm{k+1}}|s_{\mathrm{i}}^{\mathrm{k}})$ for connected states $s_{\mathrm{j}}^{\mathrm{k+1}}$ and $s_{\mathrm{j}}^{\mathrm{k}}$ is 0, if the channel memory represented by $s_{\mathrm{j}}^{\mathrm{k+1}}$ and $s_{\mathrm{j}}^{\mathrm{k}}$ contradict, or some constant c ($0 < \mathrm{c} \leq 1$). For clarity, all edges for which $p(s_{\mathrm{j}}^{\mathrm{k+1}}|s_{\mathrm{i}}^{\mathrm{k}}) = 0$ have been removed in Figure 3. Therefore the edges into state $s_{\mathrm{i}}^{\mathrm{k+1}}$ are weighted by $-\log(p(y[\mathrm{k}+1]|s_{\mathrm{j}}^{\mathrm{k+1}})$. The solution to problem 1 corresponds to the minimum cost path through all N sets of states which can be found using the Viterib algorithm.

Viterbi Algorithm: (TODO: Decide if this is clear enough)

---

given $p(y[\mathrm{k}]|s_{\mathrm{i}}^{\mathrm{k}})$, $p(\mathbf{s}_{\mathrm{l}}^{\mathrm{k+1}}|\mathbf{s}_{\mathrm{j}}^{\mathrm{k}})\ \forall\ \mathrm{k} \in 1..N$ and $\forall\ \mathrm{j} \in 1..L$.
for $\mathrm{k} = 1...N$
  for $\mathrm{i} = 1...L$
    Let $\mathrm{cost}_{\mathrm{i}}^{\mathrm{k}} = \underset{\mathrm{j}\in\{1...L\}}{\mathrm{argmin}} \left(\mathrm{cost}_{\mathrm{j}}^{\mathrm{k\text{-}1}} - \log(p(y[\mathrm{k}]|\mathbf{s}_{\mathrm{i}}^{\mathrm{k}})p(\mathbf{s}_{\mathrm{i}}^{\mathrm{k}}|\mathbf{s}_{\mathrm{j}}^{\mathrm{k\text{-}1}}))\right)$
return detected transmission $\hat{\mathbf{x}}$ corresponding to trellis path of $\underset{\mathrm{j}\in\{1...L\}}{\mathrm{argmin}}\ \mathrm{cost}_i^N$

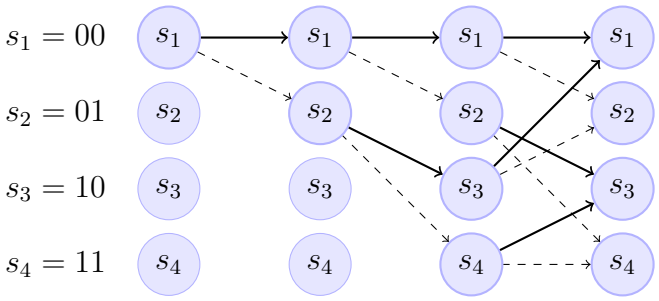

Figure 3: MLSE decoding for $\mathcal{A} = \{0,1\}$ and $L = 2$. Each state $s_1 ... s_4$ represents a possible $\mathbf{x}_{\mathrm{i-L+1}}$.

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 $|\mathcal{S}|$.

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 represent channel with certain kinds randomness *independent* of the transmitted symbols. In this case, the model is constrained to channels whose state satisfies the Markov property

$$p(\mathbf{s}_{\mathrm{l}}^{\mathrm{k}}|\mathbf{s}_{\mathrm{j}}^{\mathrm{k\text{-}1}}) = p(\mathbf{s}_{\mathrm{l}}^{\mathrm{k+1}}|\mathbf{s}_{0}^{\mathrm{k\text{-}1}}, \mathbf{s}_{k+1}^{\mathrm{N}}). \tag{2}$$

An example of such a channel is the same LTI system described above, but with a two-state noise floor satisfying the Markov property.

Give Example

Lastly, this model can also represent the case in which a channel $s[k]$ 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 $\mathbf{x}$ 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 $\mathbf{x}_{\mathrm{i-L+1}}^{\mathrm{i}}$, as was the case for the basic LTI channel.

0.4.2 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 $p(y_{\mathrm{i}}|s_{\mathrm{k}})$. Rather than estimate this function directly, we use Bayes' theorem and instead estimate the individual terms of

$$p(y[\mathrm{k}]|s_{\mathrm{i}}^{\mathrm{k}}) = \frac{p(s_{\mathrm{i}}^{\mathrm{k}}|y[\mathrm{k}])p(y[\mathrm{k}])}{p(s_{\mathrm{i}}^{\mathrm{k}})}.$$

- $p(s_{\mathrm{i}}^{\mathrm{k}}|y[\mathrm{k}])$: For a channel with finite states $\{s_1, s_2, ..s_{\mathrm{L}}\}$, this probability mass function can be estimated using a neural network for classification. Each training data pair includes a single received pilot symbol $y[\mathrm{k}]$ (the network input) and a classification into the known state $s_{\mathrm{i}}^{\mathrm{k}} \sim \mathbf{x}_{\mathrm{k-L+1}}^{\mathrm{k}}$ (the network output). 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 $p(s_{\mathrm{i}}^{\mathrm{k}}|y[\mathrm{k}])$ for all $s_{\mathrm{k}} \in S$. Figure 4 depicts the neural network for a channel with four states. The shown network has a total of edges (model parameters) to be optimized during training. For a detailed review of classification using neural networks, see [2, Ch. 5].

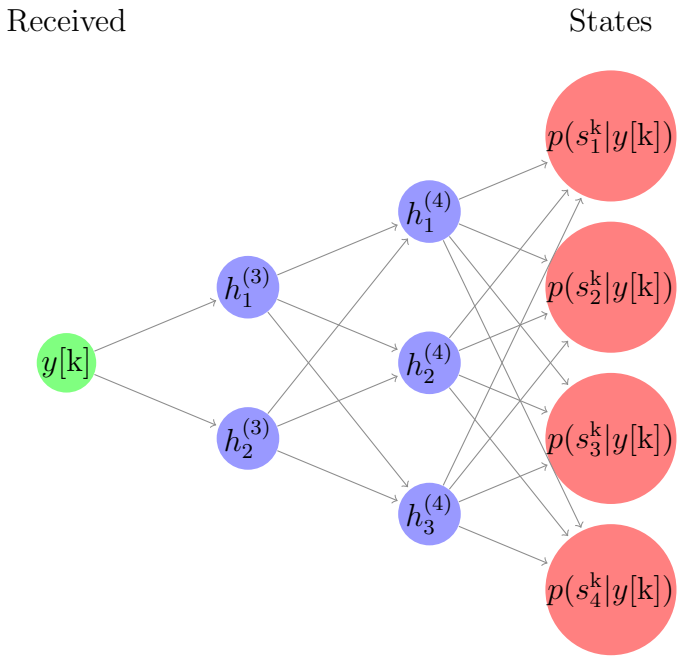

Figure 4: A fully-connect, feed-forward neural network architecture for classification of received symbols $y[\mathrm{k}]$ into four channel states.

- $p(y[\mathrm{k}]) = \sum_{s_\mathrm{i} \in \mathcal{S}} p(s_\mathrm{i}, y[\mathrm{i}])$: This term is found by marginalizing the joint probability of channel state s_i^k and received signal $y[\mathrm{k}]$ over all channel states $\mathcal{S}$. As $p(y[\mathrm{k}])$ 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 $y[\mathrm{k}]$ relative to other symbols $y[\mathrm{j}]$, $j \neq k$. For example, if $y[\mathrm{k}]$ is received under improbable noise and channel state, this term would reduce the overall impact of $y[\mathrm{k}]$ on the final path chosen through the graph. In the case of equiprobable channel states without receiver noise $p(y[\mathrm{k}])$ 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 [5], 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. (Figure 5)

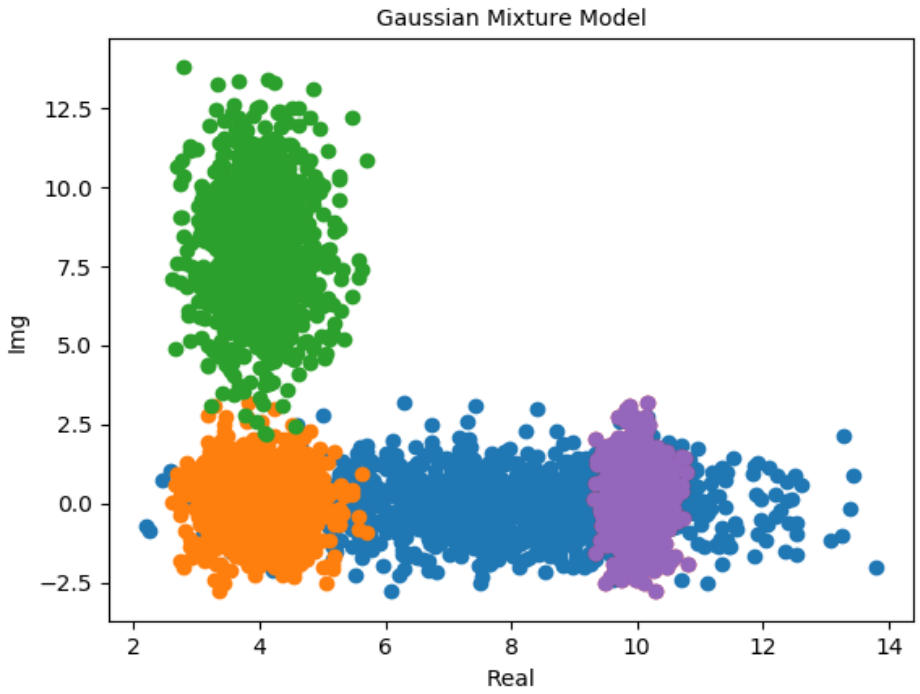

Figure 5: A mixture of complex, Gaussian sources that can be estimate using the Expectation Maximization algorithm

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

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

0.4.3 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. (might add latency).

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 particular, this is possible if the system is such that some channel states are redundant. The following, relevant example with state redundancy is posed and one potential method of reducing the number of states is then considered.

Consider a received signal from an LTI communication channel

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

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

For $\mathbf{a} = [a[1]...a[5]] = [1, 0, .2, .2, .4]$ ($\|\mathbf{a}\|_2^2 = 1$), the received signal is shown in Figure **??**. Desipite having channel memory $\mathrm{L} = 5$, there are fewer than the potential $|\mathcal{A}^{\mathrm{i}}_{\mathrm{i-L+1}}| = 2^5$ 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 and thus 16 of the potential 32 states are removed. Further, as there are two taps of value 0.2 and $\mathcal{A} = \{-1, 1\}$, when $[x[k-2], x[k-3]] = [-1, 1]$ and $[x[k-2], x[k-3]] = [1, -1]$ these map to the same state.

Pre-filtering section. Regardless, this sort of pre-filtering will require some sort of channel estimate which will require some model. Can also mention that this might introduce latency into the system unlike the design used here.

An alternative, data-driven approach to exploiting state redundancy is to cluster the original states into a set of $k < |\mathcal{A}|$ 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}, $\mathrm{i} \in \{1..N\}$
1. Label x_i with the index c of the closest centroid using $\|x_{\mathrm{i}} - \mathrm{L}_c\|_2^2$.

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 and is only performed once before deploying this detector scheme. (TODO still needs further clarification)

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

0.4.4 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, mixture model equalization.

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

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

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

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

the optimal LMMSE equalizer is [6]

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

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

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

and

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

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 therefor require more training data. This trade-off comes with the potential to learn much more complicated channels with the neural network than with the LMMSE equalizer.

In the results below, the theoretical symbol error rate curve is also included as a reference and is defined by

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

0.5 Results

0.5.1 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 well-know results with other sources.

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 $\rightarrow$ Negative Log-Likelihood)
- Training Data Examples: 5000
- Neural Network Updates: Admm [4] 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 (CHECK)
  - Equalizer Channel Memory: Always provided with the true channel memory length.

0.5.2 Simulation Results

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

We first evaluate the detector performance using the Linear Time-Invariant channel

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

with impulse response $a[l] = ...$ Simulation results are shown in fig 6

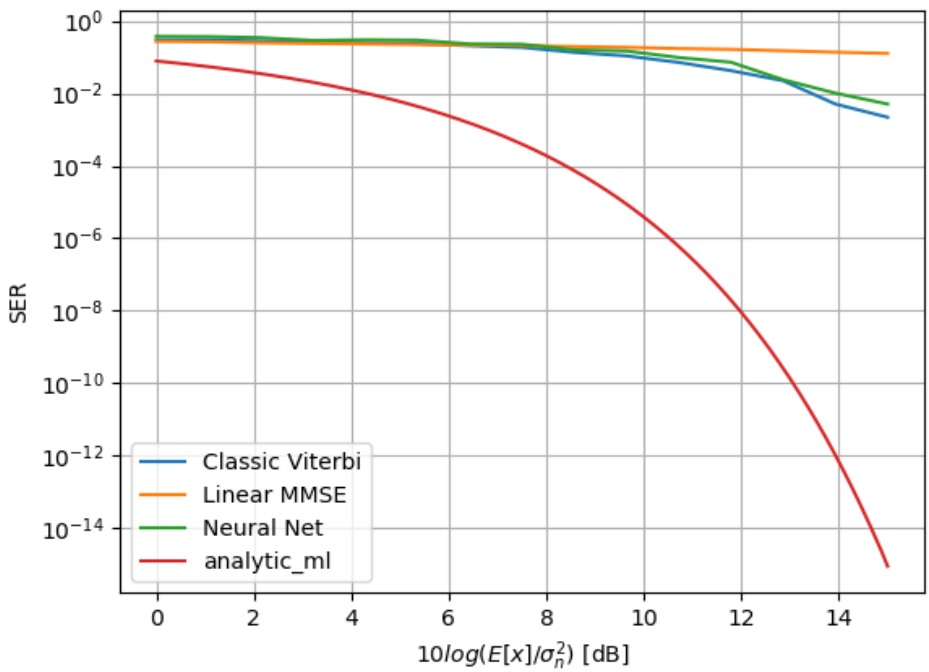

Figure 6: Detection performance over LTI channel with AWGN

LTI + Quantizer Channel

To evaluate decoder performance in the non-linear channel setting, we apply a quantizer, Quant(), to the output of the previous LTI + AWGN channel.

$$y[\mathrm{k}] = textQuant(\sum_{\mathrm{l}=1}^{\mathrm{L}} a[l]x[\mathrm{k}-l]) + n[\mathrm{k}].$$

Here we evaluate performance under two levels of quantization severity: rounding channel output down to a given number of decimal places . Figure 7 depicts the quantizer and its impact on the received symbols from the LTI channel. The resulting performance is seen in Figure 8. Notice that this is essentially a reduction of the original channel states as there is now redundancy that was not previously present.

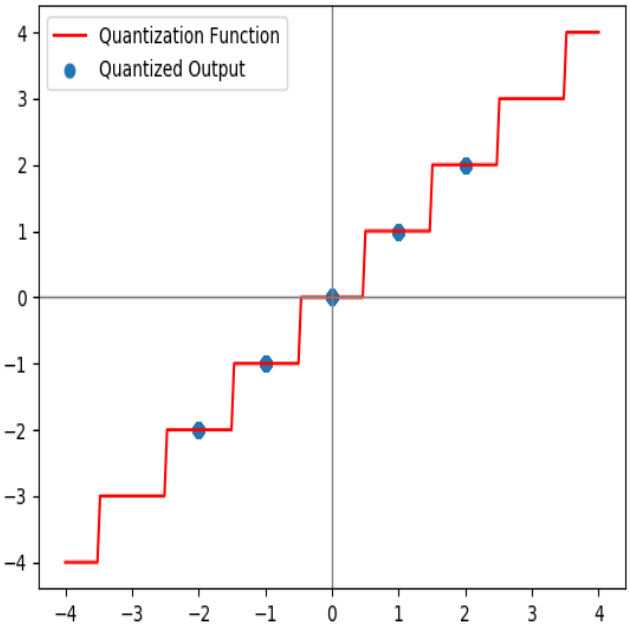

Figure 7: Quantized output of LTI channel overlayed on the applied quantization function. TODO decide on axis titles

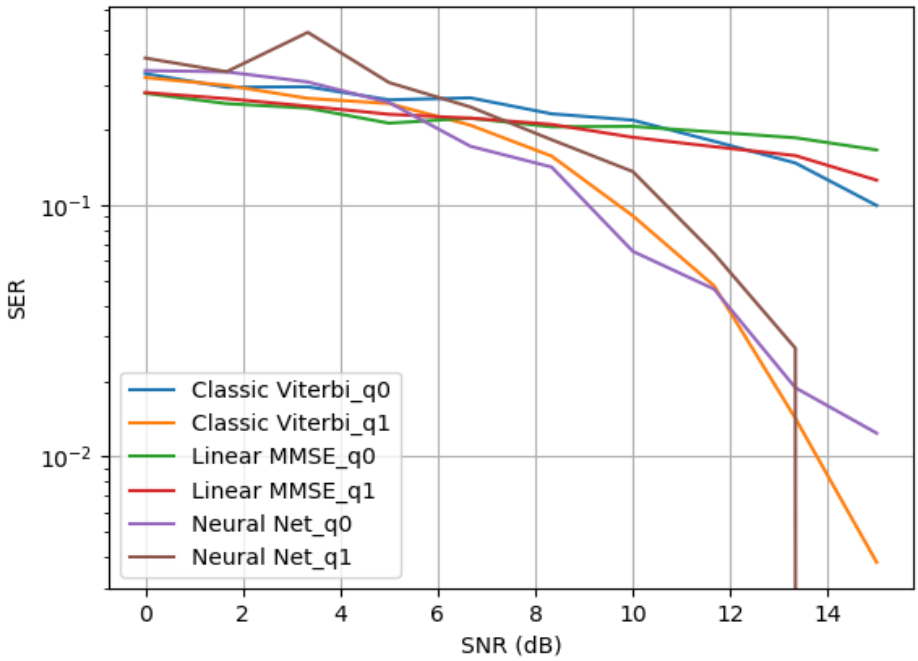

Figure 8: Detection performance over LTI channel with AWGN and quantization

molecular communication Channel

Lastly, the performance of the neural network based detector is evaluated data collected from a molecular communication channel. ... Explain if there are any results to report.

0.6 Conclusion

The proposed detection method from [7] 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 [8]

these estimate metrics have been used for minimize bit error rate estimation using the BCJR algorithm.

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.

General method extensions:

Bibliography

[1] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] Vahid Jamali, Arman Ahmadzadeh, Wayan Wicke, Adam Noel, and Robert Schober. Channel modeling for diffusive molecular communication—a tutorial review. *Proceedings of the IEEE*, 107(7):1256–1301, 2019.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Andrew Ng. Cs229 lecture notes. *CS229 Lecture notes*, 1(1), 2000.

[6] John G Proakis and Dimitris G Manolakis. *Introduction to digital signal processing*. Prentice Hall Professional Technical Reference, 1988.

[7] Nir Shlezinger, Yonina C Eldar, Nariman Farsad, and Andrea J Goldsmith. Viterbinet: Symbol detection using a deep learning based viterbi algorithm. In *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2019.

[8] Nir Shlezinger, Nariman Farsad, Yonina C. Eldar, and Andrea J. Goldsmith. Data-driven factor graphs for deep symbol detection, 2020.

[9] J-J Van De Beek, Ove Edfors, Magnus Sandell, Sarah Kate Wilson, and P Ola Borjesson. On channel estimation in ofdm systems. In *1995 IEEE 45th Vehicular Technology Conference. Countdown to the Wireless Twenty-First Century*, volume 2, pages 815–819. IEEE, 1995.