ASC Major Project

Neural Network Based Sequence Detection Over Molecular Communication 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 7, 2020

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 7, 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-based estimate of 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 supervised learning method. The neural network-based estimation is tested in simulations and shown to perform well for relevant channels.

Glossary

Abbreviations

Operators

$\|\| \cdot \|\|$	Euclidean Norm
$p(\cdot)$	Probability
$p(\cdot\|\cdot)$	Conditional Probability
sign($\cdot$)	Sign Operator
floor($\cdot$)	Floor Operator

Notation

The following notation is used through this work.

- Vectors are denoted by bold-font lower-case letters ($\mathbf{x}$) and are assumed column vectors.
- $x \sim \mathcal{N}(0,1)$ denotes a Gaussian distributed random variable x with mean 0 and variance 1.
- 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 (A).
- The set A^N denotes the set of all sequences length N for which each element is selected from A with replacement.
- The cardinality of a set A is denoted by $|A|$.
- The set of complex and real numbers are denoted by $\mathscr{C}$ and $\mathscr{R}$ respectively.
- $\operatorname*{argmin}_x f(x)$ is the value of variable x which minimizes a function $f(x)$.

Chapter 1

Report

1.1 Introduction

The general communication channel is equivalent to a conditional probability distribution function (pdf). The 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 $p(\mathbf{x}|\mathbf{y})$ [10]. The receiver is informed of a specific 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 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 Section 1.3.4. 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 [8] and [9], a neural network-based estimation of the channel is shown to be an effective method for detection with incomplete channel information for LTI 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 non-linear and random components of the channel that may be either independent or dependent on 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 [4].

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 individual elements of **x** are chosen from a finite symbol alphabet A such that the kth symbol in **x**, $x[\mathrm{k}]$, is selected from A.

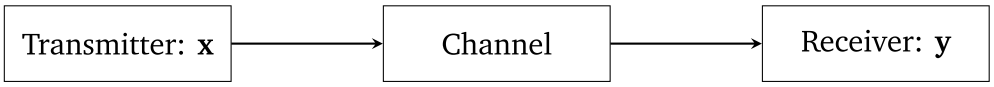

Figure 1.1: The point-to-point communication system.

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

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

with independent $n[\mathrm{k}] \sim \mathcal{N}(0,1)$. Allowing for $f_{\mathrm{k}}(\cdot)$ to be a general and potentially random function for time index k, each received symbol $y[\mathrm{k}]$ is potentially a function of all transmitted information **x**. By assuming all $n[\mathrm{k}]$ to be independent, an orthogonal filtering at the receiver of the modulated information **x** is implied.

We define the signal to noise ratio (SNR) at the receiver as

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

We now consider specific cases for the function $f_{\mathrm{k}}(\cdot)$.

In the first communication channel 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 $[a[0], a[1]...[\mathrm{L}-1]]$ (the channel impulse response).

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

In order to preserve the defined SNR, we consider the case in which the impulse response vector is unit normalized such that $||\mathbf{a}||^2 = 1$. This is also enforced in simulations discussed in Section 1.4.2.

We also consider the case in which the output from an arbitrary communication channel is the input to a quantizer defined by

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

and

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

for a given number of quantization levels with an upper and lower saturation level. A quantization function with 11 quantization levels and lower/upper saturation levels -0.5/+0.5 respectively is shown in Figure 1.2.

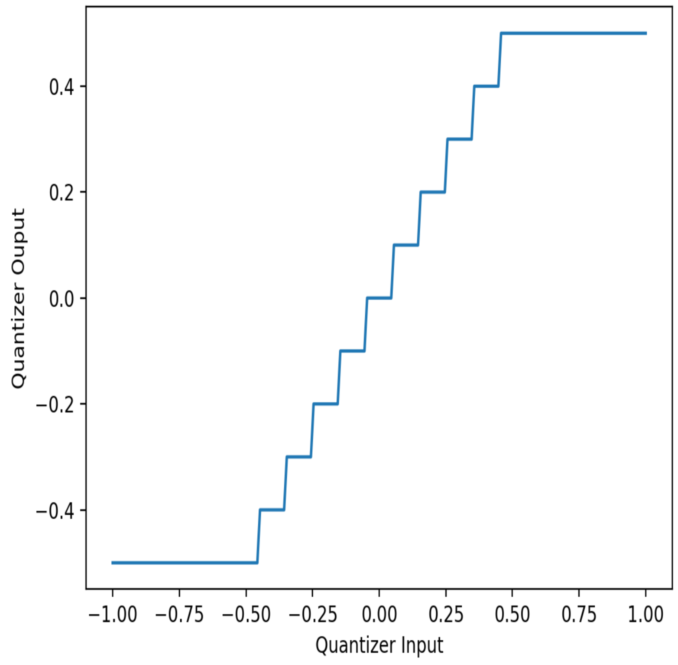

Figure 1.2: A Quantization function with with 11 levels of quantization, lower saturation level -0.5 and upper saturation level 0.5.

After quantization the received signal is then given by

$$y[\mathrm{k}] = \text{quant}(f_{\mathrm{k}}(\mathbf{x}) + n[\mathrm{k}]). \tag{1.2}$$

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 introduced. With this foundation in place, the data-driven, neural network-based estimator is then incorporated into the optimization algorithm to give a neural network-based equalizer. Extending on the neural network-based equalizer, 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 $\{s_1, s_2 ... s_\mathrm{L}\} = S$. With knowledge of the potential states $\{s_1, s_2 ... s_\mathrm{L}\}$ but not of the sequence of realized states, $[s[1], s[2] ... s[\mathrm{N}]] \in S^N$, corresponding to the observed sequence, $[y[1], y[2] ... y[\mathrm{N}]]$, the MAP problem is formalized as

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

Using Bayes' theorem, gives

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

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

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

will be used in the remainder. Noting that the size of the search space $|S^N|$ grows exponentially in N, the length of the received sequence, a method for reducing the search complexity is now introduced.

For the LTI communication channel with impulse response length L, defined by equation (1.1), each state $s_i \in \{s_1, s_2 ... s_\mathrm{L}\}$ represents a transmitted sequence $[x[\mathrm{i}-\mathrm{L}+1] ... x[\mathrm{i}]]$, implying a specific transmit sequence memory. Therefore, a state sequence $\mathbf{s}$ is only possible if the states do not have contradicting channel memory. This can be formalized by noticing that for the LTI system, the sequence of states satisfies the property

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

or equivalently, that the channel state sequence satisfies the Markov property given by

$$p(s[\mathrm{k}]|\,[s[\mathrm{k}-1]\ldots s[1]]) = p(s[\mathrm{k}]|s[\mathrm{k}\text{-}1]).$$

For this channel, the MAP sequence estimation is given by

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

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

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

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

This is illustrated in the following example.

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

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

The resulting state graph (or trellis) is depicted in Figure 1.3.

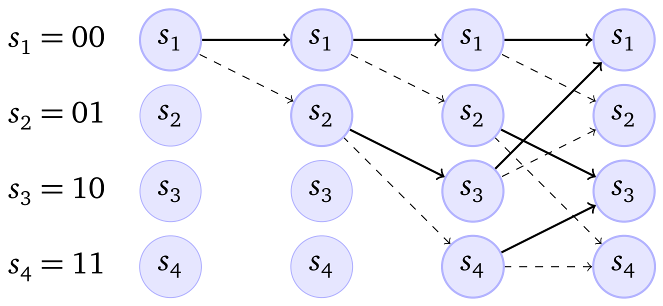

Figure 1.3: MAP decoding for and LTI channel with $\mathscr{A} = \{0, 1\}$ and channel memory $L = 2$. Each state $s_1\ldots s_4$ represents a possible transmit symbol sequence $[x[\mathrm{i}-1], x[\mathrm{i}]]$.

Assuming equiprobable $x[\mathrm{k}] \in A$, 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 $s_{\mathrm{i}}[\mathrm{k}]$ is 0, if the channel memory represented by $s_{\mathrm{j}}[\mathrm{k+1}]$ and $s_{\mathrm{i}}[\mathrm{k}]$ contradicts, and otherwise $p(s_{\mathrm{j}}[\mathrm{k+1}]|s_{\mathrm{i}}[\mathrm{k}])$ is 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 1.3. The solution to Problem

1.3 corresponds to the minimum cost path through all N sets of states, which can be found using the Viterbi algorithm.

Viterbi Algorithm

```
given $p(y[k]|s_i[k])$, $p(s_i[k+1]|s_j[k])$ $\forall$ k $\in$ 1... $N$, $\forall$ i $\in$ 1... $L$ and $\forall$ j $\in$ 1... $L$.
initialize $\text{cost}_i[0] = 0$, $\forall$ i $\in$ 1... $L$
for $k = 1 \ldots N$
    for $i = 1 \ldots L$
        Let $\text{cost}_i[k] = \underset{j \in \{1 \ldots L\}}{\text{argmin}} \left(\text{cost}_j[k\text{-}1] - \log(p(y[k]|s_i[k])p(s_i[k]|s_j[k\text{-}1]))\right)$
return the sequence of states, s, corresponding to the path of $\underset{i \in \{1 \ldots L\}}{\text{argmin}}\ \text{cost}_i[N]$
```

Note that the Viterbi algorithm is *linearly* complex in the length of the sequence **s** (N in the algorithm above) but *exponentially* complex in the number of channel states $|S|$ at each time index.

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 sequence satisfies the Markov property.

This model can also represent the case in which a channel state, $s_i[k]$, does not uniquely identify a symbol that was transmitted in the corresponding time index, $x[k]$. 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 further in Section 1.3.3.

Before moving on, an example is posed for which the states of this framework represent more than just the transmitted sequence memory. We allow for the receiver from equation 1.1 to have a noise floor such that

$$y[k] = \sum_{l=0}^{L-1} a[l]x[k-l] + n[k] + b[k](\text{noise floor}) \tag{1.4}$$

still with $n[k] \sim \mathcal{N}(0,1)$ and the Bernoulli random variable $b[k]$ describing a fluctuating noise floor at the receiver. In this case, the states from the original LTI system are extended to indicate the noise floor state for the time index of each received symbol $y[k]$. When $b[k]$ is a Bernoulli random variable, this results in doubling the number of states in each step of the Viterbi algorithm.

In the remainder of this background, we consider the case in which the channel state is determined entirely by $[x[\mathrm{i}-\mathrm{L}+1]...x[\mathrm{i}]]$.

1.3.2 ViterbiNet

Despite the complexity reduction of the Viterbi Algorithm for MAP detection, weighting the edges of the trellis requires knowledge of the channel probability distribution $p(y[\mathrm{k}]|s_{\mathrm{i}}[\mathrm{k}])$. 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 $\{s_1, s_2, \ldots s_{\mathrm{L}}\}$, 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 the probability of being in each state $s_{\mathrm{i}}[\mathrm{k}] \sim [x[\mathrm{i}-\mathrm{L}+1]...x[\mathrm{i}]]$ (the network output). The true state is known for pilot symbols. By 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 $s_{\mathrm{i}} \in S$. 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 [3, Ch. 5].

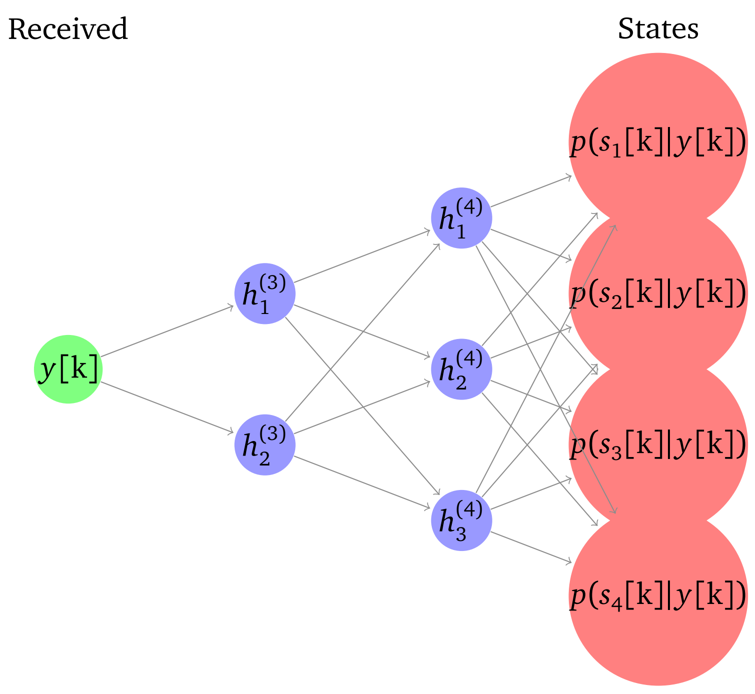
Received
States
$y[\mathrm{k}]$
$h_1^{(3)}$
$h_2^{(3)}$
$h_1^{(4)}$
$h_2^{(4)}$
$h_3^{(4)}$
$p(s_1[\mathrm{k}]|y[\mathrm{k}])$
$p(s_2[\mathrm{k}]|y[\mathrm{k}])$
$p(s_3[\mathrm{k}]|y[\mathrm{k}])$
$p(s_4[\mathrm{k}]|y[\mathrm{k}])$

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

- $p(y[\mathrm{k}]) = \sum_{s_\mathrm{i} \in S} p(s_\mathrm{i}, y[\mathrm{i}])$: This term is found by marginalizing the joint probability of channel state $s_\mathrm{i}[\mathrm{k}]$ and received signal $y[\mathrm{k}]$ over all channel states in 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 the receiver noise. The Expectation Maximization algorithm, detailed in [6], along with a set of channel output data, can be used to optimize the parameters of this model. Unlike parameterizing the neural network, this algorithm does not require labeled data from pilot sequences. An example in which the mean of 16 Gaussian sources calculated for the output of an LTI channel with 16 states (Gaussian sources) is shown in Figure 1.5. It is seen in Figure 1.5 that some of the sources result in closely overlapping output observed at the receiver.

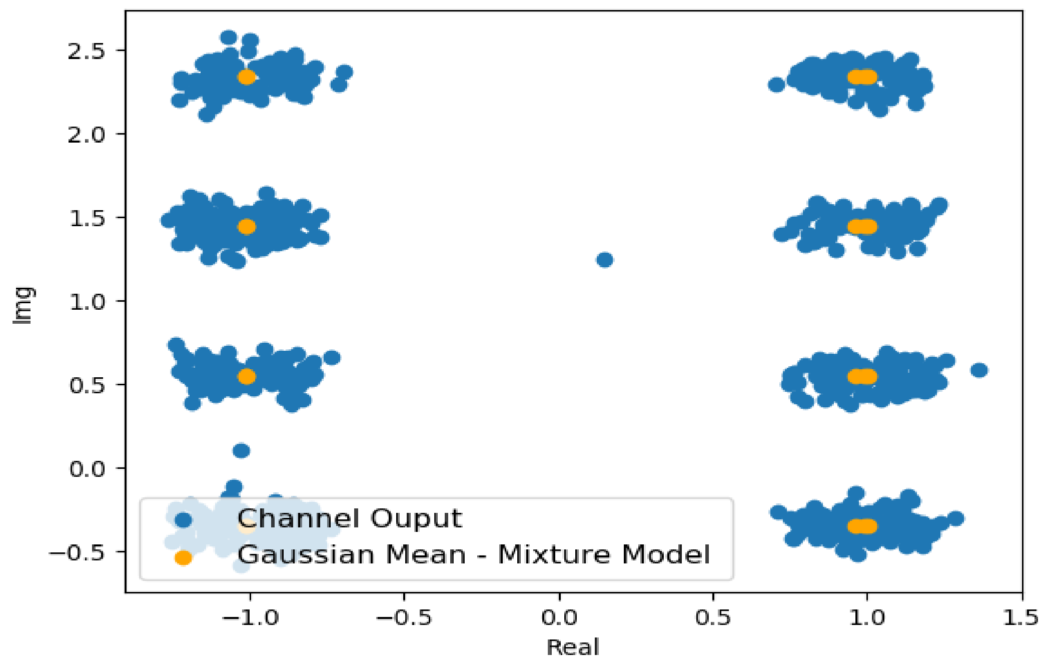

Figure 1.5: A mixture of complex, Gaussian sources and the averages of the Gaussian sources predicted using the Expectation Maximization algorithm

- $p(s[\mathrm{k}])$: For the case in which channel states are equiprobable, such as when s_k corresponds to $[x[\mathrm{i}-\mathrm{L}+1]...x[\mathrm{i}]] \in A^\mathrm{L}$, this term will not influence the final path.

Combining these terms and their impact on the original $p(y[\mathrm{k}]|s_\mathrm{i}[\mathrm{k}])$, each edge weight used in the Viterbi algorithm essentially gives the probability of simultaneously arriving at a state $s_\mathrm{i}[\mathrm{k}]$ conditioned on the previous state $s_\mathrm{i}[\mathrm{k}\text{-}1]$ 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.

1.3.3 A Reduced-State ViterbiNet

While the Viterbi Algorithm complexity scales exponentially with 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 prefiltering the incoming signal so as have a minimum-phase representation of the channel, as in [2]. Such methods are, however, dependent upon an estimate of the channel, which implies an assumed underlying channel model. In this section, we consider a method for complexity reduction without prefiltering the estimated channel.

The following, illustrative example proposes a channel whose states, as seen by the receiver, contains redundancy. A method for reducing the number of states based on this observed redundancy is then considered.

The received signal from an LTI communication channel

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

with $A = \{-1, 1\}$, $n[k] \sim \mathcal{N}(0,1)$ 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 $|A^5| = 2^5$ 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 not observed at the receiver. Additionally, because $[a[2], a[3]] = [0.2, 0.2]$, the cases in which $[x[k-2], x[k-3]] = [-1, 1]$ and $[x[k-2], x[k-3]] = [1, -1]$, will 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 $k < |S|$ 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** a set of N training points $x[\text{i}]$, $\text{i} = [1...N]$.
**select** random, initial locations of centroids $L_c$, for $c = [1... \text{K}]$.
**for** a chosen number of iterations:
    **for** training data point $x[\text{i}]$, $\text{i} = [1...N]$:
        1. Label $x[[i]]$ with the index, c, of the closest centroid using $||x[\text{i}] - \text{L}_\text{c}||_2^2$.
    **for** centroid $L_c$, $c = [1... \text{K}]$:
        1. Move $L_\text{c}$ to the average location of all training points $x[[i]]$ with label c.
**return** centroid locations, $L_c$, for $c = [1... \text{K}]$.
```

With a pilot sequence training set, a mapping of the original states into the new $k < |S|$ states is performed by finding which of the k new states (clusters) includes the most trainings examples from the known, original state s_i for $\text{i} = [1... L]$. The Viterbi algorithm can then be performed to find the MAP state sequence $\mathbf{s}$. An important implementation consideration for this clustering approach is now discussed.

If the channel has memory L > 1, some of the channels states represent different transmit sequence history. If states with contradicting transmission history are clustered together, the choice for the transmitted symbol at the index of contradiction 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 procedure, the fraction of training samples from each original state placed into each reduced state is recorded to give the likelihood of each symbol in the transmitted symbol memory for each of the reduced states. After finding the MAP state sequence **s**, the probability of each symbol in A is found for each time index. The most likely symbol is chosen in order to find a transmitted sequence estimate **x**.

The performance of the proposed system will depend on the whether or not the number of clusters, k, corresponds to the true number of states observed by the receiver. If k is much smaller than the number of observed states, the performance will suffer. Again considering the LTI channel with AWGN, if k is smaller than the observed number of clusters at the receiver, the equalization will be disregarding a component of the channel memory, and as a result, may perform poorly.

Before discussing the results of simulations using the neural network-based equalization schemes, a fundamental, low-complexity equalization scheme is introduced.

1.3.4 Equalization Benchmark

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

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

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

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 [7]

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

$E[\mathbf{R}_{yy}]$ 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

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

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 therefore require more training data. The need for a larger training data set, however, comes with the 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, $A = \{-1, +1\}$, scheme is simulated giving [7]

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

with r as the received signal and E_b being the energy per bit (or symbol, for BPSK) at the receiver.

1.4 Results

1.4.1 Implementation Details

This section defines the relevant implementation information for the simulations used to generate the results in Section 1.4.2.

- **Neural Network:** The neural network used in this simulation was implemented using the PyTorch machine learning library. For details regarding general neural network design, refer to [3]. The network architecture and training parameters used are as follows:
  - Architecture: Four, fully connected layers based on the network used in [8]. Layer 1 (input) has 1 node, layer 2 has 100 nodes using the hyperbolic tangent activation, layer 3 has 50 nodes using the ReLU activation, layer 4 (output) has a number of nodes corresponding to the number of channel states assumed. A softmax function taken over all output layer nodes to give a probability distribution over the channel states.

  - Loss Function: Cross Entropy.
  - Training Data Examples (Pilot Sequence Length): 5000.
  - Edge Weight Update Method: Adam with step size 10^{-2}, see [5] for details.
  - Batch Size: 1000
  - Backpropogation Updates (Epochs): 900

- **Expectation Maximization Algorithm:** A general Expectation Maximization algorithm is derived in [6]. The parameters chosen for this algorithm are as follows:
  - Mixture Model Source Types: Gaussian (based on the assumption of Gaussian receiver noise).
  - Number of Sources: This number is equal to the number of outputs chosen for the neural network and therefore corresponds to the number of channel states assumed.
  - Training Data Examples: 5000

- **k-Means Algorithm (For Reduced State ViterbiNet):** The parameters chosen for the algorithm shown in Section 1.3.3 are as follows:
  - The number of clusters to create, k: This would typically be chosen case-by-case, based on the number of channels states observed at the the receiver and performance tolerance. Here we always select to reduce the channel down to 8 states.
  - Training Data Examples: 5000

- **Linear Minimum Mean-Squared Error (LMMSE) equalizer:** The parameters to the LMMSE derivation shown in Section 1.3.4 are as follows:
  - Equalizer Channel Memory, L: This value is assumed to be known by the receiver.
  - Training Data Examples: 5000

1.4.2 Simulation Results

The following section evaluates the performance of the equalizers described in this work using simulations details in 1.4.1. 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 using complex-valued noise.

We first evaluate the detector performance using LTI channels as defined in equation (1.1). In order to compare the performance of the reduced state equalizer, we consider two cases, both with potentially 2^5 channel states. One LTI channel with state redundancy (channel impulse response $= [.9, 0, 0, .4, .7]$) (Fig. 1.6) and one without channel state redundancy (channel impulse response $= [0.9, 0.7, 0.3, 0.5, 0.1]$) (Fig. 1.7). In the case with channel redundancy, Figure 1.6, and only 2^3 states considered by the reduced-state equalizer, it is seen that both neural network-based equalizers outperform the LMMSE equalizer and that the non-reduced state neural network-based equalizer is similar to the Viterbi algorithm performed with perfect knowledge of the channel function $p(\mathbf{x}|\mathbf{y})$. Similar performance of the standard neural network-based equalizer is seen in Figure 1.7. As expected, however, the reduced state equalizer is virtually useless if the channel has more states than are accounted for in the clustering as in Figure 1.7.

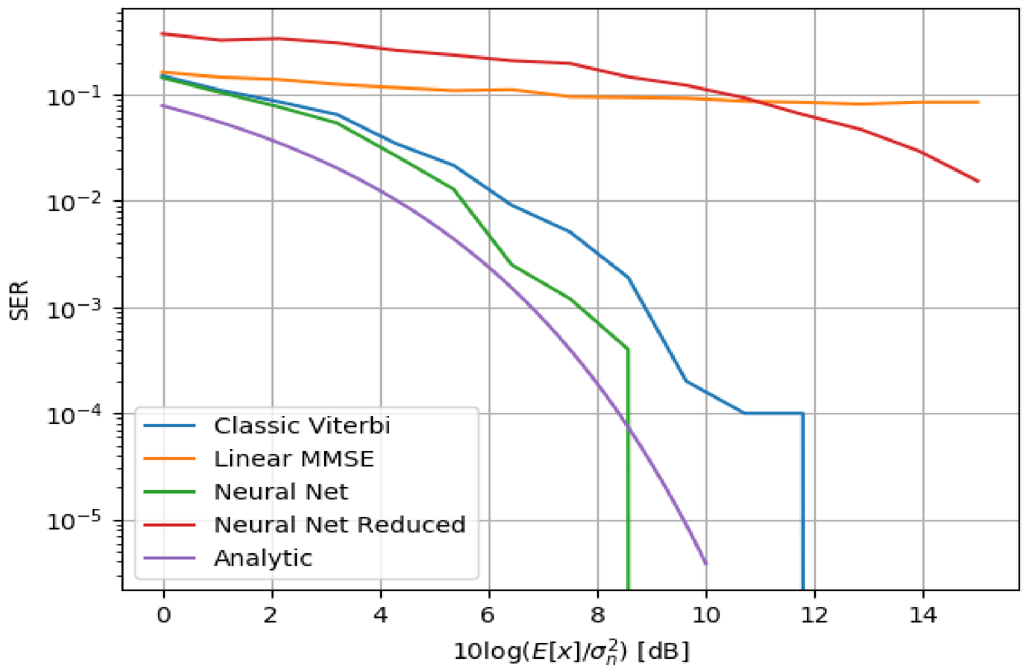

Figure 1.6: Detection performance over LTI channel with AWGN. The LTI channel has impulse response $[.9, 0, .0, .4, .7]$.

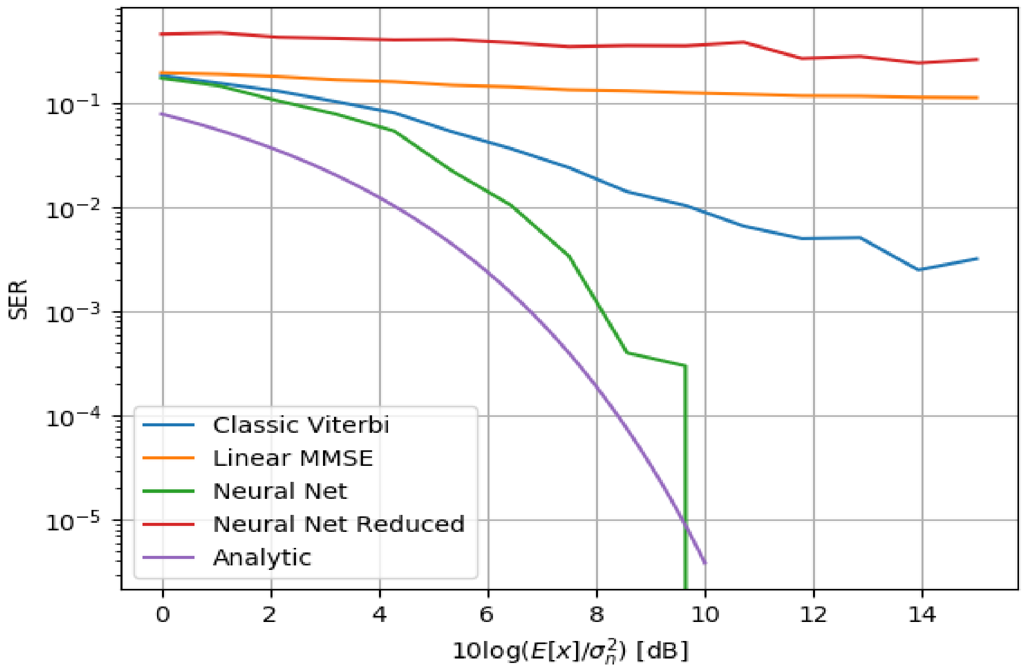

Figure 1.7: Detection performance over LTI channel with AWGN. The LTI channel has impulse response $[0.9, 0.7, 0.3, 0.5, 0.1]$.

We now consider the same LTI channels above but with receiver quantization, defined in equation (1.2) and plotted in Figure 1.2, which rounds the received signal to a single decimal place. The upper and lower saturation levels of the quantizer are chosen based on the set of training data such that no received symbols are clipped to the saturation level.

Figures 1.8 and 1.9 show the results of these simulations. While in both cases, the reduced-state neural network-based equalizer performs poorly, the standard neural-network based equalizer performs well, indicating that this method is capable of learning the distribution function $p(\mathbf{x}|\mathbf{y})$ for non-linear channels.

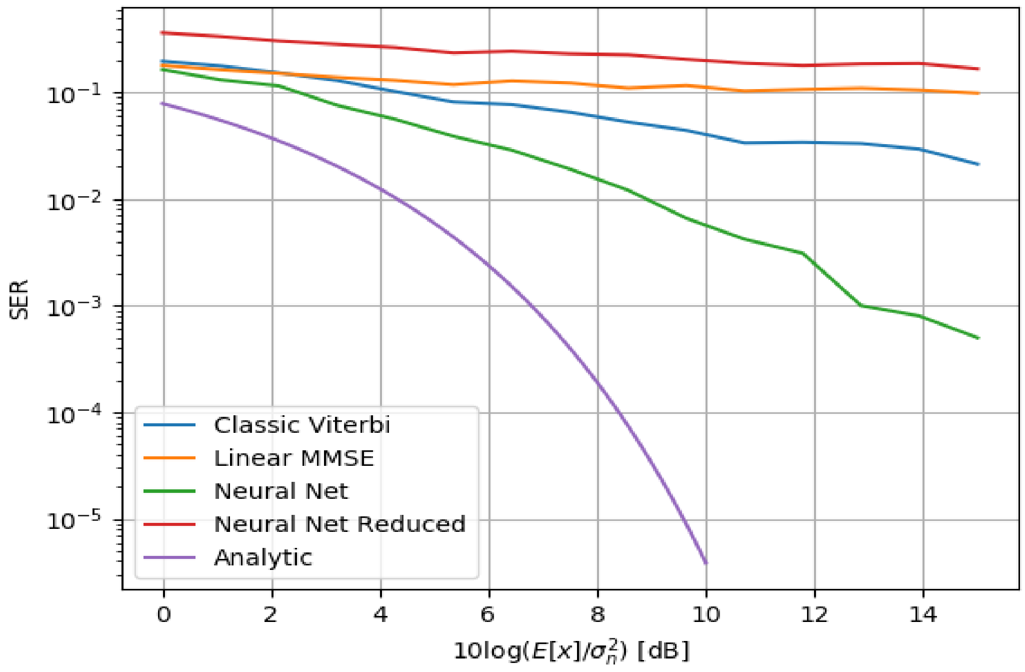

Figure 1.8: Detection performance over quantized, LTI channel with AWGN. The LTI channel has impulse response $[.9, 0, 0, .4, .7]$.

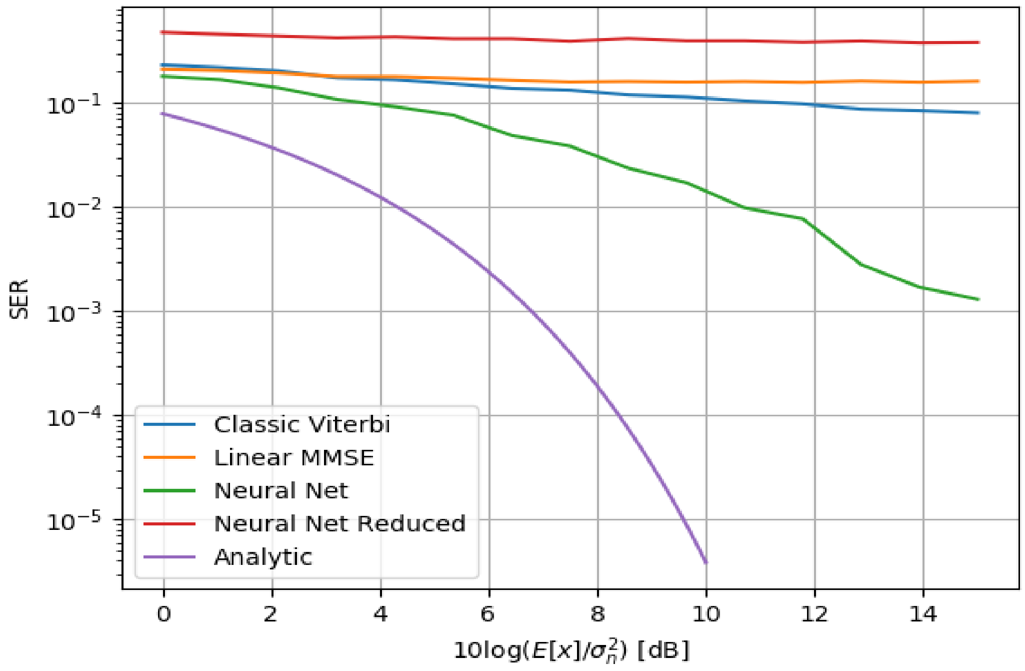

Figure 1.9: Detection performance over quantized LTI channel with AWGN. The LTI channel has impulse response $[0.9, 0.7, 0.3, 0.5, 0.1]$.

1.5 Conclusion

In this work, a framework is developed in order to define the class of channels to which the channel estimation techniques shown in the work from [8] can be applied. Our framework shows that the methods in [8] can be used in order to learn channels with addition types of randomness. The framework also outlines a method for complexity reductions in the resulting Viterbi Algorithm. In the simulation work, the results of [8] are extended to show that the neural network-based equalizer is capable of channel estimation for non-linear channels and may be a viable method for estimating channels in the molecular communications domain.

In future work, this general framework should be explored in related detection algorithms such as those used in [9]. Further simulation should also be performed for evaluating the detection performance over channels with other non-linearities and random behavior.

Bibliography

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

[2] Wolfgang H Gerstacker, Frank Obernosterer, Raimund Meyer, and Johannes B Huber. An efficient method for prefilter computation for reduced-state equalization. In *11th IEEE International Symposium on Personal Indoor and Mobile Radio Communications. PIMRC 2000. Proceedings (Cat. No. 00TH8525)*, volume 1, pages 604–609. IEEE, 2000.

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

[4] 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.

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

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

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

[8] 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.

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

[10] 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.