Undergraduate Project Report

ON

" APPLICATIONS OF MACHINE LEARNING TO COMMUNICATION SYSTEMS"

Submitted to

Department of Electrical Engineering ,
Indian Institute of Technology -Bombay

BY

Sujay Mundada

UNDER THE GUIDANCE OF

Prof. Kumar Appaiah

Acknowledgements

I’m profoundly thankful to professor Kumar Appaiah for helping and guiding me throughout this project

ABSTRACT

We present and discuss several novel applications of machine learning (ML) for the physical layer, by interpreting a communications system as an autoencoder and analyse the role of convolutional neural networks and deep neural networks on raw IQ samples for modulation classification.

Blind Pre-processor Modulation Classifier

We tend to build a deep neural network to help classify modulation used to transmit the digital data before processing stage by using supervised learning and tend to show that they outperform traditional classification techniques based on expert features.

Modelling Communication Channels as Autoencoders

We address the problem of learning an efficient and adaptive physical layer encoding to communicate binary information over an impaired channel. In contrast to traditional work, we treat the problem an unsupervised machine learning problem .We tend to reconstruct the wireless channel model as an end-to-end model that seeks to jointly optimize transmitter and receiver components in a single process.

Keywords: Supervised Learning, Autoencoder, Unsupervised Learning, Modulation, Physical Layer

Contents

List of Figures

1 Introduction

Radio communication/wireless communication is all about finding efficient ways to transfer information over a channel which can be as simple as corruption with a Gaussian noise to more complex channels exhibiting features like multi-path fading, impulsive noise and so on.

Theoretically we often know the upper or lower bounds on various quantities like achievable capacity and information density vs bit error rates for given modulation schemes, bandwidths and signal to noise ratios etc. In practice we have been able to much closer to them but this usually involves a complex and expensive hardware support and DSP software tuning which may not be feasible in everyday practice.

By taking approach of unsupervised and supervised learning we sought to optimize the reconstruction of a communication channel or classify modulation schemes at the preprocessor level.

1.1 Potential of ML for the physical layer

ML algorithms could provide improvement over existing physical layer algorithms. Most common models in wireless systems involve linear, stationary mathematical models and have Gaussian statistics. Real-life systems however have a lot of imperfections and non-linearities. For this reason deep learning bases systems can be optimized for a particular hardware configuration. Moreover most of the communication system models involve individual optimization of the various blocks

involved in the transmission of the signal like source, channel, modulation, encoding, equalization. Although this system has given us very efficient models but it not entirely clear that making such individual components efficient will lead to best end-to-end performance. Attempts to jointly optimize these components as it done in machine learning however will guarantee us the best end-to-end performance.

1.2 Project Goal & Specification

The goal of our project to look closely at 2 applications of machine learning in communication systems. The first one involves the classification of modulation at the pre-processor level, which will help us reduce the overhead in transmission of signals which usually involves encoding the type of modulation in the transmitted signal as well. The second application involves using unsupervised learning to model an end-to-end communications systems by optimization of both the trasnmitter and receiver model as one deep neural network that can be trained as an autoencoder to minimize the reconstruction loss and show that this achieves equivalent BLER compared to the traditional methods of hamming encoding.The beauty of this approach is that it can even be applied to channel models and loss functions for which the optimal solutions are unknown.

2 | Basics of Machine Learning

2.1 Types of machine learning

- *MachineLearning*
  - Supervised learning
    - Classification
    - Regression
  - Unsupervised learning
    - Discovering clusters
    - Discovering latent factors
    - Discovering graph structure
    - Matrix completion

2.2 Three elements of a machine learning model

Model = Representation + Evaluation + Optimization

2.2.1 Representation

In supervised learning, a model must be represented as a conditional probability distribution $P(y|\vec{x})$ (usually we call it classifier) or a decision function $f(x)$. The set of classifiers(or decision functions) is called the hypothesis space of the model. Choosing a representation for a model is tantamount to choosing the hypothesis space that it can possibly learn.

2.2.2 Evaluation

In the hypothesis space, an evaluation function (also called objective function or risk function) is needed to distinguish good classifiers(or

decision functions) from bad ones.

2.2.3 Loss function and risk function

In order to measure how well a function fits the training data, a **loss function** $L : Y \times Y \to R \geq 0$ is defined. For training example (x_i, y_i), the loss of predicting the value $\widehat{y}$ is $L(y_i, \widehat{y})$.

The following is some common loss functions:

1. 0-1 loss function
   $$L(Y, f(X)) = I(Y, f(X)) = \begin{cases} 1, & Y = f(X) \\ 0, & Y \neq f(X) \end{cases}$$
2. Quadratic loss function $L(Y, f(X)) = (Y - f(X))^2$
3. Absolute loss function $L(Y, f(X)) = Y - f(X)$
4. Logarithmic loss function
   $L(Y, P(Y|X)) = -\log P(Y|X)$

   The risk of function f is defined as the expected loss of f:

$$R_{\text{exp}}(f) = E\left[L(Y, f(X))\right] = \int L(y, f(x))\, P(x, y)\mathrm{d}x\mathrm{d}y \tag{2.1}$$

which is also called expected loss or **risk function**.

The risk function $R_{\text{exp}}(f)$ can be estimated from the training data as

$$R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) \tag{2.2}$$

which is also called empirical loss or **empirical risk**.

You can define your own loss function, but if you're a novice, you're probably better off using one from the literature. There are conditions that loss functions should meet:

1. They should approximate the actual loss you're trying to minimize. As was said in the other answer, the standard loss functions

for classification is zero-one-loss (misclassification rate) and the ones used for training classifiers are approximations of that loss.

2. The loss function should work with your intended optimization algorithm. That's why zero-one-loss is not used directly: it doesn't work with gradient-based optimization methods since it doesn't have a well-defined gradient (or even a subgradient, like the hinge loss for SVMs has).

   The main algorithm that optimizes the zero-one-loss directly is the old perceptron algorithm.

2.2.4 Optimization

Finally, we need a **training algorithm**(also called **learning algorithm**) to search among the classifiers in the the hypothesis space for the highest-scoring one. The choice of optimization technique is key to the **efficiency** of the model.

2.3 Machine Learning Libraries

In recent times, numerous tools and algorithms have emerged that make it easy to build and train large NNs. Tools to deploy such training routines from high level language to massively parallel GPU architectures have been key enablers. Among these are Caffe, MXNet, TensorFlow, Theano, and Torch (just to name a few), which allow for high level algorithm definition in various programming languages. We have used tensorflow to code the NNs in this paper.

3 | Communication Channel as Autoencoder

3.1 Introduction

In its simplest form, a communications system consists of a transmitter, a channel, and a receiver, as shown in Fig. 3.1.

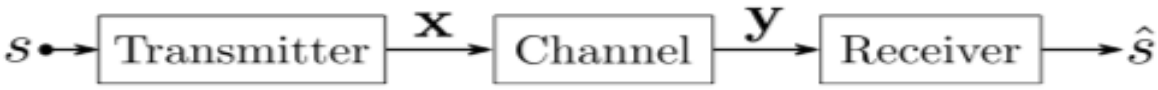

Figure 3.1: Simple Communication Channel Model

The transmitter wants to communicate one out of M possible messages s : M = 1,2,...,M to the receiver making n discrete uses of the channel. To this end, it applies the transformation f to the message s to generate the transmitted signal x = f(s).The communication rate of this communications system is R = k/n, where k = log2(M). In the sequel, the notation (n,k) means that a communications system sends one out of M = 2k messages (k bits) through n channel uses.
From a DL point of view, this simple communications system can be seen as a particular type of autoencoder. Typically, the goal of an autoencoder is to find a low-dimensional representation of its input at some intermediate layer which allows reconstruction at the output with minimal error. In this way, the autoencoder learns to non-linearly compress and reconstruct the input. In our case, the purpose of the autoencoder is different. It seeks to learn representations x of the messages s that are robust with respect to the channel impairments mapping x to y (i.e., noise, fading, distortion, etc.), so that the transmitted

message can be recovered with small probability of error. In other words, while most autoencoders remove redundancy from input data for compression, this autoencoder often adds redundancy, learning an intermediate representation robust to channel perturbations. The simple scheme of our autoencoder is shown in fig. 3.2

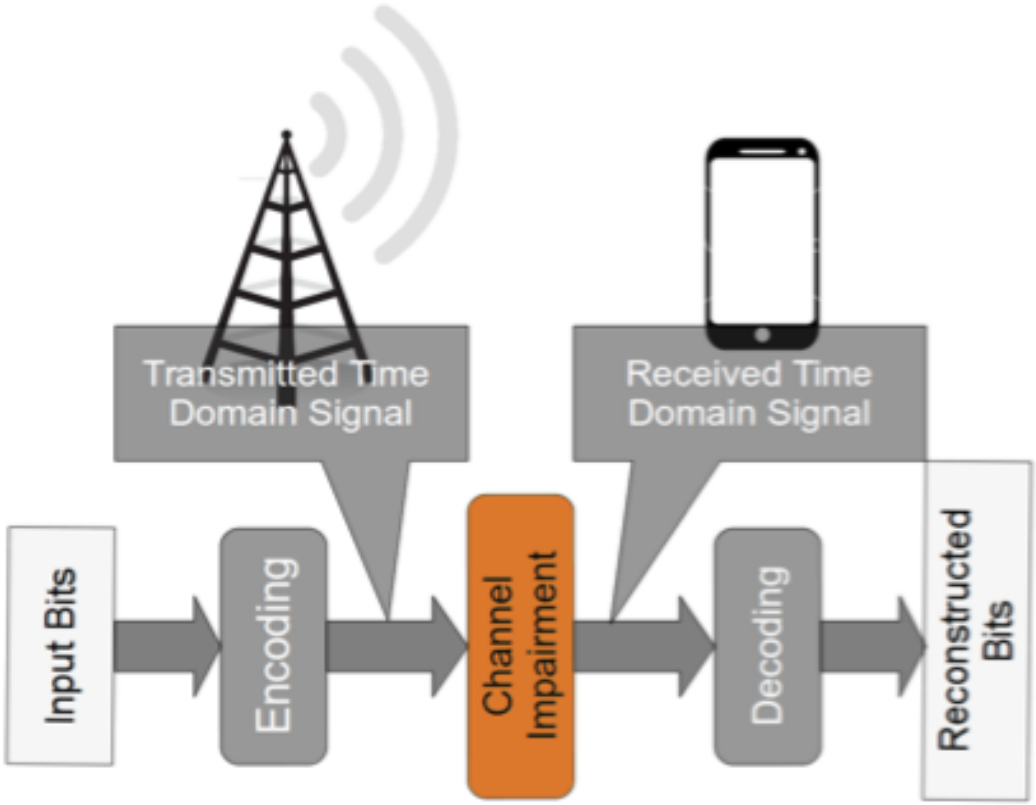

Figure 3.2: Modelling the channel as an Autoencoderl

3.2 Implementation

We constructed a new gaussian noise layer that adds a random normal noise to the transmitted signal.
The network works as is shown if fig. 3.3 and the parameters of the network built are shown in fig. 3.4 where M = 16 and n =128.

Layer	Output dimensions
Input	M
Dense + ReLU	M
Dense + linear	n
Normalization	n
Noise	n
Dense + ReLU	M
Dense + softmax	M

Figure 3.3: Autoencoder Network

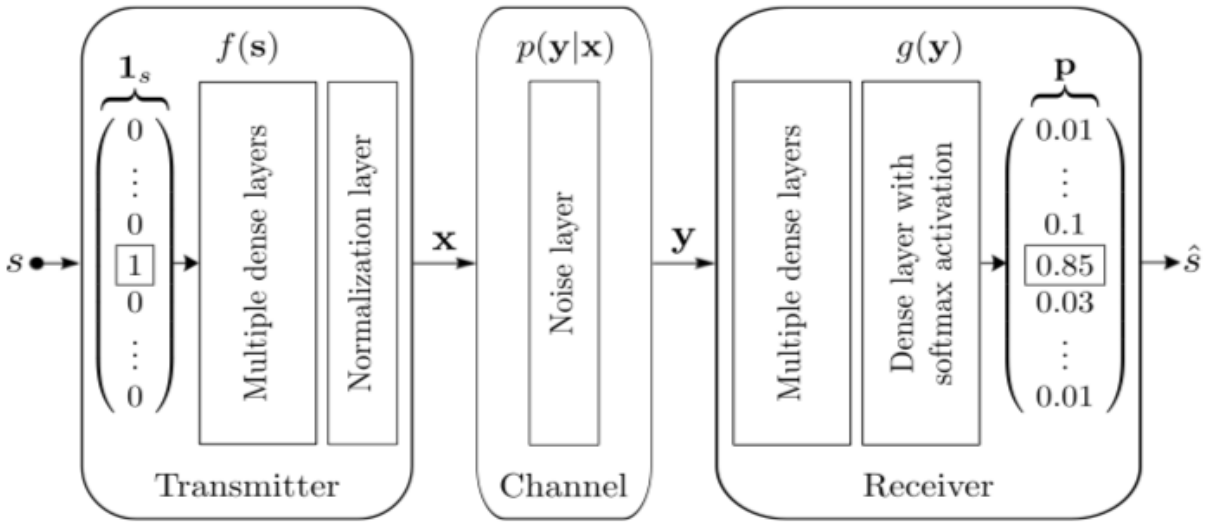

Figure 3.4: Network Parameters

3.3 Problems Faced

The training was done and tested at a particular SNR but the results were not matching as proposed in the paper. We expected the performance of our autoencoder to match with that of hamming encoder with MLD decoding as shown in the below graph.

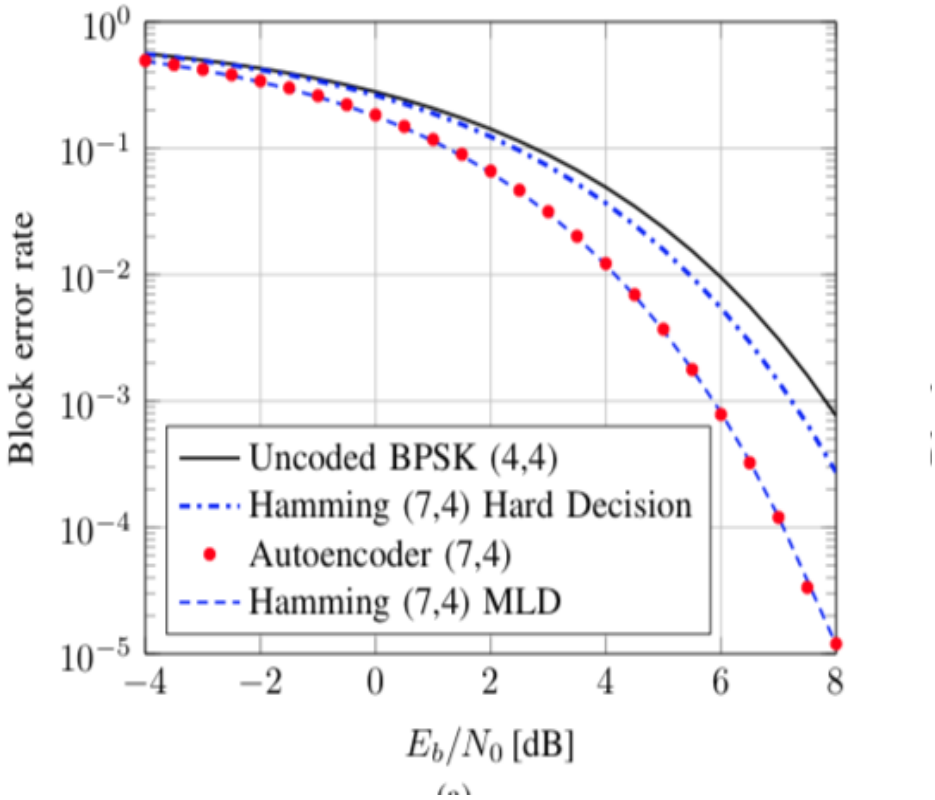

Figure 3.5: Modelling the channel as an Autoencoderl

4 | Blind Pre-Processor Classifier

4.1 Introduction

Modulation classification can be defined as the process of determining the modulation scheme of a noisy signal from a given set of possible schemes. This process has many applications in wireless communications, including in autonomous multimode and software defined radios. In most scenarios of interest, the difficulty in performing modulation classification is due primarily to the fact that classifiers operate with no or incomplete knowledge of the fading experienced by the signal and the distribution of the noise added in the channel. This is because a receiver typically has to first classify the received signal before it can successfully acquire symbol timing and estimate the channel state.The modulation classification operation can typically be divided into two stages. In the first, referred to as the pre- processing stage, the signal is acquired and different channel state and noise distribution parameters are estimated. With the obtained estimates, the signal is then classified in the second stage using an appropriate classification algorithm.

4.2 Applications

Modulation classification was first motivated by its application in military scenarios where electronic warfare, surveillance and threat analysis requires the recognition of signal modulations in order to identify adversary transmitting units, to prepare jamming signals, and to recover the intercepted signal.

Link adaptation (LA), also known as adaptive modulation and coding (AM&C), creates an adaptive modulation scheme where a pool of multiple modulations are employed by the same system. It enables the optimization of the transmission reliability and data rate through the adaptive selection of modulation schemes according to channel conditions. While the transmitter has the freedom to choose how the signals are modulated, the receiver must have the knowledge of the modulation type to demodulation the signal so that the transmission can be successful.

An easy way to achieve that is to include the modulation information in each signal frame so that the receivers would be notified about the change in modulation scheme,and react accordingly. However, this strategy affects the spectrum efficiency and throughput due to the extra modulation information in each signal frame. In the current situation where the wireless spectrum is extremely limited and valuable, the aforementioned strategy is simply not efficient enough. For this reason, pre-processor classification becomes an attractive solution to the problem.

4.3 Implementation

We have used deep neural network(DNN) for the classification task. In DNN neurons are organized into layers: input, hidden and output. The input layer is composed not of full neurons, but rather consists simply of the record's values that are inputs to the next layer of neurons which are the I and Q components of the modulated received signal in our case. The next layer is the hidden layer. Several hidden layers can exist in one neural network, our network consists of 3 hidden layers with 256 neurons each. The final layer is the output layer, where there is one node for each class, 2 for our case. A single sweep forward through the network results in the assignment of a value to each output node which can be thought of as probability of detecting each class, and thus the record is assigned to the class node with the highest value probability.

4.3.1 Data Set Generation & Representation

The data set was generated using Octave. The data set comprises of 20000 data points each having 3 fields (I component, Q component and modulation class)

4.3.2 Training & Loss Functions

The original data set is divided into 80 percent for training and 20 percent for testing.
The batch size is 500 with 2000 iterations each and learning rate used is 0.001 .
Loss function used is cross entropy loss function.

4.4 Results & Observation

We obtained 100 percent classification for classification of QPSK vs 16QAM.
Using the channel model y= Hx + n given H we could use the same model with new training to obtain the same percentage classification. If H changes slowly with time as it happens practically we could train the old model using as fewer as 100 iterations and still obtain the same results.

4.5 Future Work

- Extend the model to together classify all the most frequently used 9 digital and 2 analog modulation techniques.
- Try to train the network for a worse case impulsive noise training data and test for the gaussian noise cases.
- Tap the individual hidden layers to find out what features are they extracting at each level.
- Use multiple samples of the same transmitted symbol and use Convolutional neural network to classify thereby using the correlation property between different samples of the same bit transmitted.
- Try to use progressive learning so that the the system to predict the change of channel matrix H

5 | Open Research Challenges

In order to compare the performance of ML models and algorithms, it is crucial to have common benchmarks and open datasets. While this is the rule in the computer vision,voice recognition, and natural language processing domains (e.g., MNIST10 or ImageNet11), nothing comparable exists for communications. This domain is somewhat different because it deals with inherently man made signals that can be accurately generated synthetically, allowing the possibility of standardizing data generation routines rather than just data in some cases. It would be also desirable to establish a set of common problems and the corresponding datasets (or data-generating software) on which researchers can benchmark and compare their algorithms. One such example task is modulation classification ; others could include mapping of impaired received IQ samples or symbols to codewords or bits.