Automated Code Design for Real-valued Channels

Tavor Baharav `tavorb` Jeffrey Barratt `jbarratt` Shane Barratt `sbarratt`

Original Research / Art Project

1 Introduction

Error Correcting Codes (ECCs) form a set of techniques for robustly sending data over unreliable channels. Classical ECCs consider channels that map elements of Cartesian products of finite fields to another (or the same) element in that field. Robustness can be measured by how many changes or erasures the code can tolerate in the worst case, or by the probability of error in Shannon's model, when errors are assumed to be random. There are many notions of optimality, governed by possibility and impossibility results based on the rates of codes versus their distance.

In this paper we instead consider real-valued channels, *i.e.*, channels that map between real vector spaces. In some applications, *e.g.*, communication and storage, channels that have this form more naturally model the actual underlying communication channel. As the number and diversity of mediums over which we perform communication and storage increase, it will become more and more important to develop specialized codes for such mediums.

We propose an automated solution for developing such codes, given a model of the channel, which can be constructed by system identification. We demonstrate the utility of our method by applying it to several well-known channels, and showing that it can find reasonably good codes. Our report also includes many compelling visualizations of the codes that we discover; it turned out to be an art project as well!

2 Related work

Coding theory, like most other fields, has felt the impact of machine learning. Work is currently being done on deep learning-based approaches to communication (see [KJR+18] for a survey), so it is interesting to compare our simpler optimization methods to the more complicated schemes based on deep learning. For another paper regarding the use of deep learning in coding theory, we note that some work has also been done for joint source-channel coding, which appears to perform very well at shorter block lengths [BKG18]. Some channel coding work similar to ours has recently been studied in [OH17].

DeepSig, a company trying to apply machine learning to signal processing, has studied many aspects similar to the results seen in this paper. They have studied multiple-input and multiple-output (MIMO) processing using deep learning in [OEC17]. Also, O'Shea applied deep learning to radio communication which is simply another type of channel in [ORC17]. Further results in the application of machine learning to signal processing from this company will likely continue to arise as time progresses.

3 Design problem

Encoder/decoder design problem. We consider the problem of designing encoders and decoders for sending messages over noisy real-valued vector channels. That is, given a message $x \in \{1, \ldots, k\}$ that comes from a distribution $p(x)$, we encode the message as the codeword $c = E(x)$, where

$$E : \{1, \ldots, k\} \to \{c \mid \|c\|_2 \leq P\}$$

is a (norm-limited) encoding or embedding function and $P > 0$ is the maximum allowed norm. The codeword c is then sent over a random channel $C : \mathbf{R}^n \times \mathcal{W} \to \mathbf{R}^n$ (assumed to be differentiable in its first argument), and the decoder receives $\tilde{c} = C(c, \omega)$, where $\omega \in \mathcal{W}$ is random and comes from the distribution $p(\omega)$. The corrupted codeword is decoded as $\hat{x} = D(\tilde{c})$, where $D : \mathbf{R}^n \to \{1, \ldots, k\}$ is a decoding function. We assume that we know the form of C, know the distribution of $p(x)$, and can sample from $p(\omega)$.

The goal is to minimize the probability of error by optimizing the encoding and decoding functions, or to solve the optimization problem

$$\underset{E,D}{\text{minimize}} \quad \underset{x,\omega}{\mathbf{E}} \, \mathbf{1}[D(C(E(x), \omega)) \neq x]. \tag{1}$$

We will see that this description encompasses many interesting error correction design problems over real-valued channels, for different forms of C. This could also be a weighted loss, *e.g.*, if the binary string 00 is decoded as 01 it results in lower loss than it if had been decoded as 11. In this paper we consider non-weighted losses.

Generality. The problem that we have described encompasses many types of channels. In analog communication, the "channel" C could include the signal path through the hardware at the transmitter and the receiver (which could realistically be nonlinear) and the communication channel itself. It also includes, *e.g.*, complex channels, since a complex channel over $\mathbb{C}^n$ can be converted to a channel over $\mathbf{R}^{2n}$, and channels defined in the frequency domain.

Difficulty. This optimization problem is evidently difficult, since the optimization variables are functions and the objective is nonlinear and nonconvex. Therefore, as a practical matter, we will only be able to solve this problem approximately using a heuristic.

Proposed solution method. We propose to represent E as a parameterized composition of nonlinear functions, replace the indicator function in the objective with a smooth loss function, and optimize the objective using the projected decoding method. That is, we let $E(x = i) = (\theta_E)_i$ (the ith column of θ_E) for some $\theta_E \in \mathbf{R}^{n\times k}$. We also replace the decoding function with $D_{\theta_D} : \mathbf{R}^n \to \Delta_k$, where Δ_k is the K-dimensional probability simplex. We relax the nonconvex 0-1 loss with a convex relaxation, in particular the cross entropy loss, which has the form

$$l(\hat{x}, x = i) = -\log \hat{x}_i.$$

We assume that the outputs of D and E are differentiable in θ_D and θ_E respectively. We optimize the composite parameter $\theta = (\theta_D, \theta_E)$. This results in the finite-dimensional optimization problem

$$\underset{\theta}{\text{minimize}} \quad \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} p(x^{(i)}) p(\omega^{(i)})\; l(D_{\theta_D}(C(E_{\theta_E}(x^{(i)}), \omega^{(j)})), x^{(i)}), \tag{2}$$

where $x^{(1)}, \ldots, x^{(N)} \sim p(x)$ and $\omega^{(1)}, \ldots, \omega^{(M)} \sim p(\omega)$. We can (approximately) solve this problem using projected descent, since we can compute the gradient of the objective with respect to θ_D and θ_E by the chain rule. After a gradient step, we project each column of θ_D onto $\{c \mid \|c\|_2 \leq P\}$. We will solve the problem multiple times with random initializations and select the best final θ in terms of the true objective. The quality of the θ that we find will be measured by the its performance in terms of the objective in (1), and proves that the performance level is achievable (since we have restricted the feasible set of (1)).

Implementation details. In our experiments, the decoder has the form

$$D(\tilde{c}) = \phi(A_3 \max(A_2 \max(A_1 \tilde{c} + b_1, 0) + b_2, 0) + b_3),$$

where $A_1 \in \mathbf{R}^{128\times n}$, $b_1 \in \mathbf{R}^{128}$, $A_2 \in \mathbf{R}^{128\times 128}$, $b_2 \in \mathbf{R}^{128}$, $A_1 \in \mathbf{R}^{k\times 128}$, $b_1 \in \mathbf{R}^k$ and the function $\phi : \mathbf{R}^k \to \Delta_k$ has the form

$$\phi(x)_i = \frac{\exp(x_i)}{\sum_{i=1}^{k} \exp(x_j)}.$$

The parameters of the decoder are $\theta_D = (A_1, A_2, A_3, b_1, b_2, b_3)$.

We implemented the ideas described in this paper in the automatic differentiation framework PyTorch [PGC+17]. Our implementation is available as an attachment to the submission of this paper. We ran the experiments on a GTI Nvidia 1080 TI GPU using 32-bit floating numbers (floats), which led to significant speed-ups (20 times or more) as opposed to the CPU. The API exposes a method

`get_encoder_decoder(k, n, P, channel)`,

which takes the number of messages, the number of dimensions in the channel, the maximum power constraint, and the channel model. The channel model is a function that takes in a

batch of codewords $C \in \mathbf{R}^{N \times n}$ and outputs a sample of the corrupted codewords $\tilde{C} \in \mathbf{R}^{N \times n}$. For example, an AWGN channel can easily be materialized with the code snippet:

```
channel = lambda c: c + 0.2 * torch.randn(*c.shape).
```

4 Noise models

The first channel model that we consider is the additive Gaussian channel, which has the form

$$C(c, \omega) = c + \omega,$$

where $\omega \sim \mathcal{N}(0, \Sigma)$, and $\Sigma \succ 0$ is the covariance matrix. When $\Sigma = \sigma^2 I$ for some $\sigma > 0$, the channel is referred to as an additive white Gaussian noise (AWGN) channel. It is relatively trivial to derive good encoder/decoder pairs for a Gaussian channel. However, applying our procedure to these channels when $n = 2$ and visualizing the results serves as a good sanity check of the procedure.

AWGN channel. We begin with a concrete example of the procedure applied to an AWGN channel; the resulting codewords and decoding regions are displayed in figure 3a. The decoding regions roughly correspond to the Voronoi diagram defined by the points under the ℓ_2-norm, which is what an optimal decoder would have done. The encoder looks roughly like a *quadrature amplitude modulation* (QAM) constellation, as expected.

Error probability versus rate. In Fig. 1 we plot the achieved error probability versus number of messages k for a fixed channel AWGN. We see that even in a small case, the error probability increases when the number of messages on the channel k increases past the theoretical Shannon capacity of the channel of $2^{3.329} \approx 10.05$. This shows that the performance of our implementation is able to almost reach the Shannon capacity of the channel.

Colored Gaussian channel. Next we apply our procedure to a channel with colored Gaussian noise, *i.e.*, where Σ is not diagonal. We use

$$\Sigma = \begin{bmatrix} 0.01 & -0.0025 \\ -0.0025 & 0.01 \end{bmatrix}.$$

The resulting codewords and decoding regions are displayed in figure 3b. The decoding regions roughly correspond to a Voronoi diagram under the Mahalanobis distance with Σ.

We can calculate the capacity of this channel as in [CT12]. We have that

$$\Sigma = \begin{bmatrix} 0.01 & -0.0025 \\ -0.0025 & 0.01 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} .01 \begin{bmatrix} 1.25 & 0 \\ 0 & .75 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} = Q \Lambda Q^\top.$$

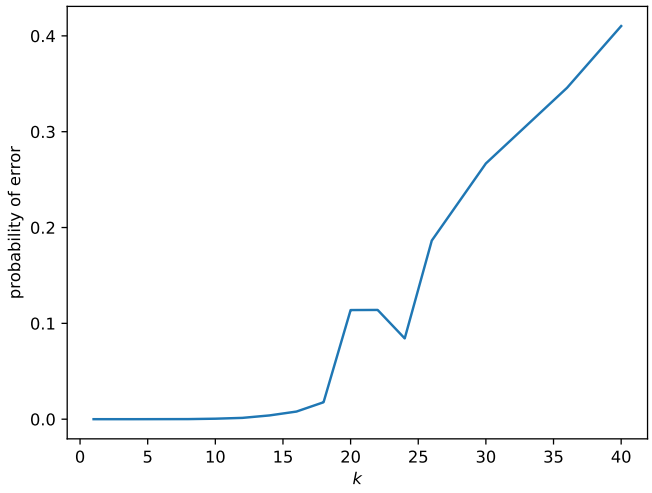


Figure 1: Error probability versus k for a fixed channel AWGN. Here $n = 2$, $\sigma = 0.2$, $P = 1$ (max power of 1), and $p(x)$ is uniform. The Shannon capacity of this model in the limit is about 10 and an increase in error is seen past that point.

With this in hand, we define $A \triangleq Q^\top K_x Q$, where we know that the output has a power constraint and so entropy will be maximized for normal output, which implies normal input. Hence we let our input be normal: $\mathcal{N}(0, K_x)$.

$$A_{ii} = (\nu - \lambda_i)^+,$$

where ν is chosen such that $\sum_{i=1}^n A_{ii} = nP$. Since our power constraint for this problem was $P = 1$,

$$2 = \sum_i (\nu - \lambda_i)^+ = (\nu - .0125)^+ + (\nu - .0075)^+ = 2\nu - .02 \Rightarrow \nu = 1.01.$$

Since $A_{11} = .9975,\ A_{22} = 1.0025$, this gives us a solution for K_x

$$K_x = QAQ^\top = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} .9975 & 0 \\ 0 & 1.0025 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}.$$

where

$$K_x + \Sigma = Q \left(A + .01 \begin{bmatrix} 1.25 & 0 \\ 0 & .75 \end{bmatrix} \right) Q^\top = Q \begin{bmatrix} 1.01 & 0 \\ 0 & 1.01 \end{bmatrix} Q^\top = \begin{bmatrix} 1.01 & 0 \\ 0 & 1.01 \end{bmatrix}.$$

Hence the output is normal: $\mathcal{N}(0, 1.01 I_2)$, and has entropy

$$\frac{1}{2} \log_2 \left(|2\pi e \cdot (1.01 I_2)| \right) = \log_2 \left(2.02 \pi e \right) \approx 4.10855.$$

This means that we can reliably transmit ≈ 17.25 messages (note this only holds in an asymptotic sense, and is just for interpretability).

Multiplicative Rayleigh channel. A multiplicative Rayleigh channel has the form

$$C(c, \omega) = \omega_1 c + \omega_2,$$

where ω_1 comes from a Rayleigh distribution and $\omega_2 \sim \mathcal{N}(0, \sigma^2 I)$. Figure 3c shows an example of a learned encoder and decoder. This channel simulates possible errors in a fiber optic network. We see the codewords taking a "wheel and spoke" format because they are each scaled by a positive real number, drawing them closer or further away from the center in the noisy channel. The center is also occupied as the codeword $[0, 0]$ does not get scaled by ω_1 and it is unlikely that ω_1 is close to 0 for other codewords under the Rayleigh distribution.

Sparse errors. Our framework naturally extends to additive channels with non-Gaussian errors. For example, the channel could have sparse errors; one model of this is

$$C(c, \omega) = c + \omega, \qquad \omega_i \sim Ber(p)\mathcal{N}(0, \sigma^2).$$

That is, with some probability p in each dimension, a normally-distributed error is added to c. Figure 3d shows an example of a learned encoder and decoder, which is quite nonintuitive. Reading tea leaves, we can conjecture that these decoding boundaries are due to the sparsity of the noise: it is more likely that the noise is only impacting the transmitted codeword in 1 dimension, and so the decoding region extends further along the axes than along a diagonal. In order to facilitate this, the points are staggered along the x and y axes.

Syndrome Decoding. This case of sparse errors traditionally leads to syndrome decoding. This can be viewed as a compressed sensing problem on the syndrome of the received codeword, where denoting H as the parity check matrix of our code, we have that

$$Hy = H(x + e) = He,$$

If we assume that e is a sparse vector, as in the above scenario, then this is exactly the compressed sensing problem we are used to, as $H \in \mathbb{R}^{n-k \times n}$ is an under determined system. If we wish to solve for the sparsest solution to this equation, we are left with the problem of

$$\min_{\tilde{e} \in \mathbb{R}^n} \|\tilde{e}\|_0 \text{ subject to } H\tilde{e} = Hy.$$

This problem is unfortunately nonconvex due to the $\|\cdot\|_0$ objective. However, if we are given that the matrix H satisfies a *restricted isometry property* (RIP) for an isometry constant $\delta_{2s} < \sqrt{2} - 1$, then as shown in [Can08], we can obtain the exact solution by instead solving a convex relaxation of this problem

$$\min_{\tilde{e} \in \mathbb{R}^n} \|\tilde{e}\|_1 \text{ subject to } H\tilde{e} = Hy,$$

under the assumption that e is s-sparse. This specific requirement can be thought of as the condition that for all subsets of at most $2s$ columns of H, the columns are nearly orthogonal [CRT06]. Formally δ_s is defined as the maximum δ_s which satisfies the below constraint:

$$(1 - \delta_s)\|x\|_2^2 \leq \|Hx\|_2^2 \leq (1 + \delta_s)\||x\|_2^2 \qquad \forall s - \text{sparse } x,$$

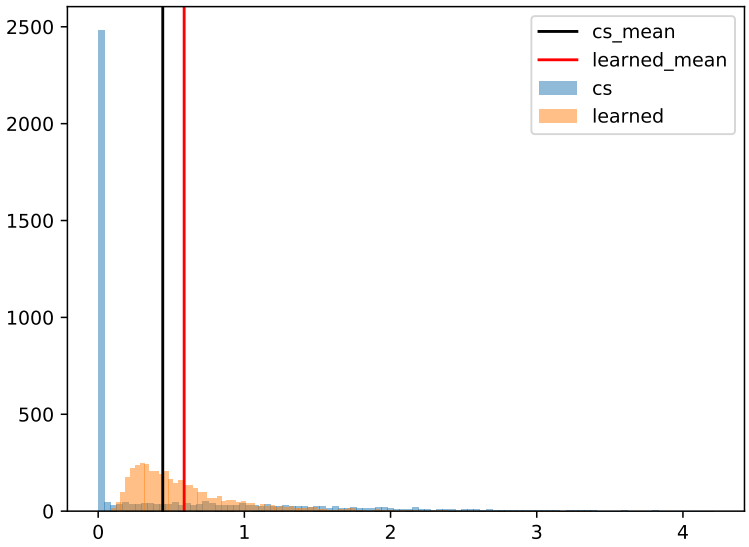


Figure 2: A histogram of mean squared error (MSE) for CS vs. learned encoder / decoder. In this example we use $C = Ber(.2) * N(0, 1)$, $n = 20$, and $k = 10$. The Y-axis is frequency out of 4096 test samples.

and this requirement is so that we are able to tell the difference between two possible error vectors e_1, e_2 which are both s-sparse (i.e. $e_1 - e_2$, a $2s$-sparse vector, does not go unnoticed via observation through H).

We simulate this numerically by generating a random parity check matrix $H \in \mathbb{R}^{n-k \times n}$, where each entry of H is drawn iid from $N(0, 1)$. We then construct our generator matrix by choosing columns so that they span the nullspace of H, i.e. that $G \in \mathbb{R}^{nxk}$ and $HG = 0$. We know that if our error is s-sparse, the singleton bound states that we need at least $n - k \geq 2s + 1$. A more detailed asymptotic analysis yields that for H Gaussian we need $n - k = O(s \log(\frac{n}{s}))$ [CRT06]. Working for finite n and k, we decided to just slightly over provision in terms of $n - k$, as this $\log(\frac{n}{s})$ is an asymptotic result.

In figure 2 we see that this ℓ_1 relaxation for compressed sensing works very well, and that over half the time it reconstructs the signal almost perfectly. However since our n was finite and our channel was $Ber(p)N(0, 1)$, we have that our vectors had sparsity according to a binomial distribution, which means that sometimes $n - k < 2s + 1$ (i.e. there is no hope of exact recovery). In these cases, we see that CS yields a very high error, having a very long tail with respect to squared error.

On the other hand, we have a learned linear encoder and parameterized nonlinear decoder. Through our work we realized that we needed to constrain the norm of our learned generator matrix, as otherwise this problem becomes simple: under this sparse Gaussian noise, $2G$ will perform better than G as the range is more spread out over $\mathbb{R}^n$ and it is relatively easier to

overcome this Gaussian noise. With this power constraint on the generator matrix, we have that the learned decoder is conservative; it will never decode the message correctly, but will always have fairly low error, as it was trained to minimize MSE loss as a surrogate for the $0-1$ loss. At the end, the MSE for Compressed Sensing is lower than that for this learned method, as we expected; there is a large body of theory on compressed sensing, and it makes sense that it would perform better than this arbitrary learned encoder.

5 Conclusion and future work

One of the main benefits of learning the encoder/decoder in the case of communications is that (in theory) it can adapt to the channel at hand and outperform hand-constructed codes. However, it is unreasonable to assume full knowledge of the channel. When the channel is not known, but we can send signals through it, the techniques described in this paper can still be used by replacing the gradient descent procedure with (guided) random search [Mat65]. This has immediate practical implications when a transmitter and receiver have a low-quality link, but are trying to bootstrap it to create a fast and reliable link. They would use the low-quality link to transmit information while learning and performing (guided) random search to learn an encoder/decoder pair. This learning paradigm may yield interesting results, as the encoder and decoder can easily spend time locally approximating gradients with respect to sent data, but communicating ground truths and results over the low-quality link are prohibitively expensive. So far, learned models as in [KJR$^+$18, BKG18] are only really promising for short block lengths. It would be interesting to see if our current learning scheme or this gradient free learning scheme would be able to perform well at longer block lengths.

References

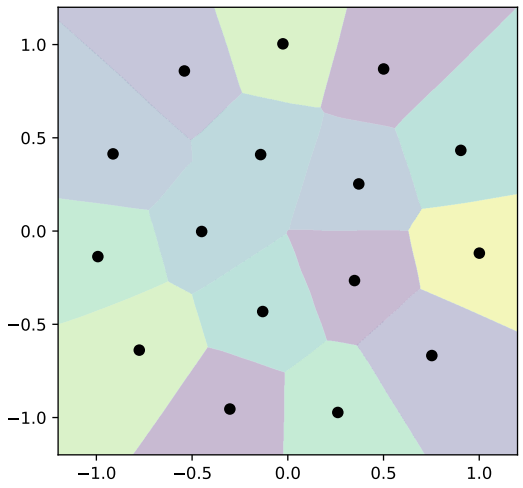
[BKG18] E. Bourtsoulatze, D. Kurka, and D. Gunduz. Deep joint source-channel coding for wireless image transmission. *arXiv preprint arXiv:1809.01733*, 2018.

[Can08] E. Candes. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 346(9-10):589–592, 2008.

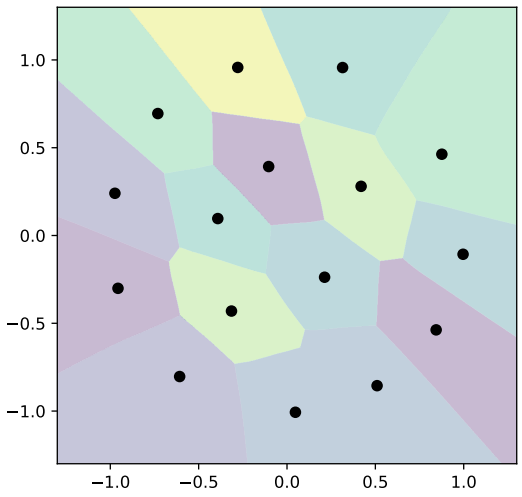
[CRT06] E. Candes, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[CT12] T. Cover and J. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

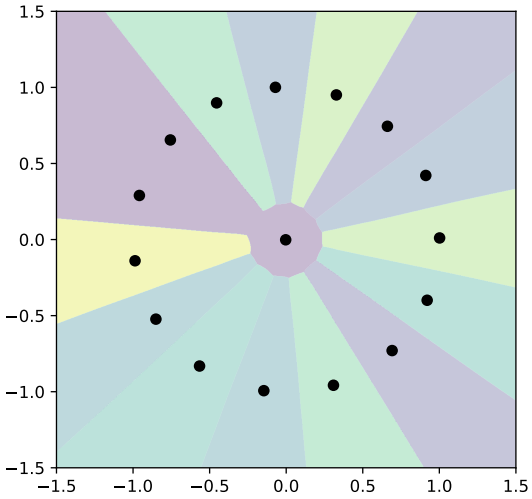
[KJR$^+$18] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath. Communication algorithms via deep learning. *arXiv preprint arXiv:1805.09317*, 2018.



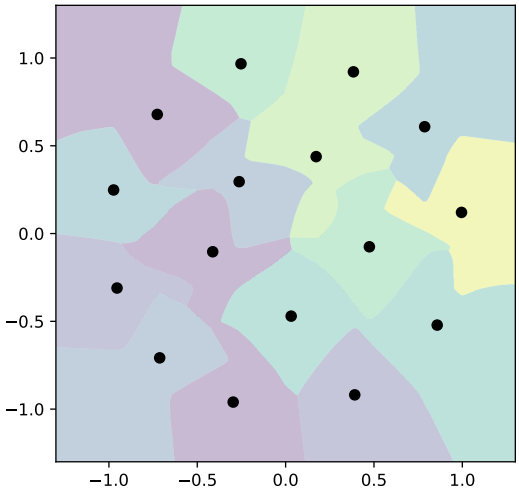
(a) AWGN.



(b) Colored Gaussian.



(c) Multiplicative Rayleigh.



(d) Additive sparse Gaussian.

Figure 3: Codewords and decoding regions for various noise patterns.

[Mat65] J. Matyas. Random optimization. *Automation and Remote control*, 26(2):246–253, 1965.

[OEC17] Timothy J. O'Shea, Tugba Erpek, and T. Charles Clancy. Deep learning based MIMO communications. *CoRR*, abs/1707.07980, 2017.

[OH17] Timothy OShea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

[ORC17] Timothy J. O'Shea, Tamoghna Roy, and T. Charles Clancy. Over the air deep learning based radio signal classification. *CoRR*, abs/1712.04578, 2017.

[PGC+17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *Neural Information Processing Systems*, 2017.