EE 321 - Fall 2020
Project 4: Classification, decision theory, and digital communications
Due Date: November 13, 2020

1 Introduction

In this project, we will look at data that is separable in some sense, and use algorithms to classify them into different groups. We will start with two simple two class classifiers, which we will derive and implement. In all cases, we will have some "training data" whose classes are known, and which we will use to build classification algorithms. The effectiveness of the algorithms will be evaluated on "testing data" that has not been seen by the algorithm. This class of algorithms is called supervised learning, in contrast to unsupervised learning, where no ground truth about the data is known.

2 Statistical Decision Theory

We consider the problem where each cluster of the data is described by a single point. Generally, this is a centroid of the cluster, and the points in the cluster are modeled as noisy measurements of the centroid of the cluster they belong to. Using this observation model, we design algorithms to classify the data. In this class of algorithms, we know the underlying statistics describing the data, making the modeling extremely powerful. Applications include RADAR, SONAR, LiDAR, Communications, Biomedicine, etc. In what follows, we will design a classification algorithm for a communications application.

3 Modulation

Consider a communications scheme where data is transmitted using several symbols, say M symbols. Assuming the symbols are transmitted on two orthogonal carriers (most commonly, a sine and a cosine), each symbol can be represented in a two-dimensional space (recall why from the PCA project) as $s_m, m = 1, 2, \dots, M$. This type of representation is called a constellation diagram (see Figure 2a for an example).

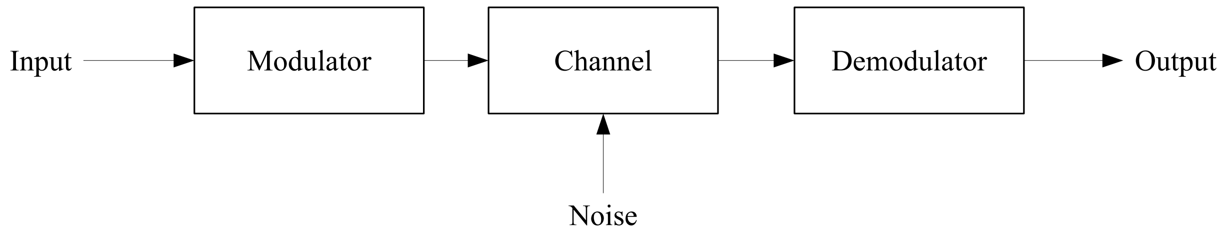


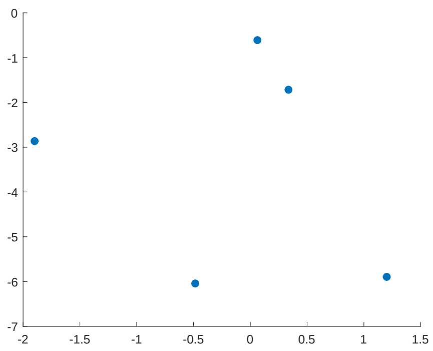
Figure 1: A communications system. An input signal is modulated and transmitted over a noisy channel. At the receiver, the corrupted data is demodulated.

We can model the transmission system as shown in Figure 1. The channel model, as this is also called, shows that a transmitted signal, s_m, is corrupted by noise, n, to give us the received signal,

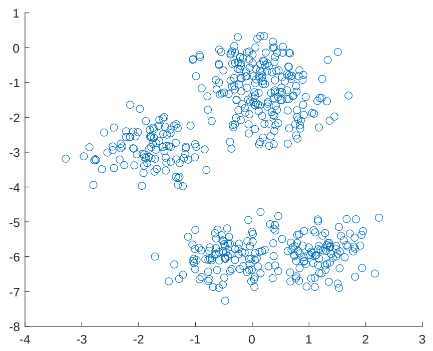
r as:

$$r = s_m + n, \tag{1}$$

where $m = 1, 2, \ldots, M$ and $n \sim \mathcal{N}(0, \sigma_n^2)$ is white Gaussian noise with zero mean and variance σ_n^2. The received data can be similarly represented using a constellation as shown in Figure 2b. The task, now, is to determine what the transmitted symbol was, given the received signal, r.



(a) Constellation diagram for a communications scheme with five symbols, represented in two dimensions.



(b) The received signal shown in the same constellation space. Notice how the noise on transmission smears each constellation center. The task at the receiver is to determine which receive point corresponds to what symbol while making minimum error.

Figure 2: A signal constellation before and after transmission.

At the receiver, we want to minimize the error, which means, we want to maximize the probability of a correct decision, and is represented as

$$\hat{s}_m = \underset{1 \le m \le M}{\operatorname{argmax}} \Pr[s_m | r], \tag{2}$$

and is called the maximum *a posteriori* or MAP estimate. Using Bayes' Rule the MAP estimate can be rewritten to give the maximum-likelihood or ML estimate:

$$\hat{s}_m = \underset{1 \le m \le M}{\operatorname{argmax}} \Pr[r | s_m], \tag{3}$$

which can further be simplified to

$$\hat{s}_m = \underset{1 \le m \le M}{\operatorname{argmax}} \, p_n(r - s_m), \tag{4}$$

where $p_n(r - s_m)$ is the Gaussian PDF with mean $r - s_m$ and variance σ_n^2, given by

$$p_n(r - s_m) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(r - s_m)^2}{\sigma_n^2}\right). \tag{5}$$

After some algebra, it can be shown that (4) reduces to

$$\hat{m} = \underset{1 \le m \le M}{\operatorname{argmin}} \left\| r - s_m \right\|. \tag{6}$$

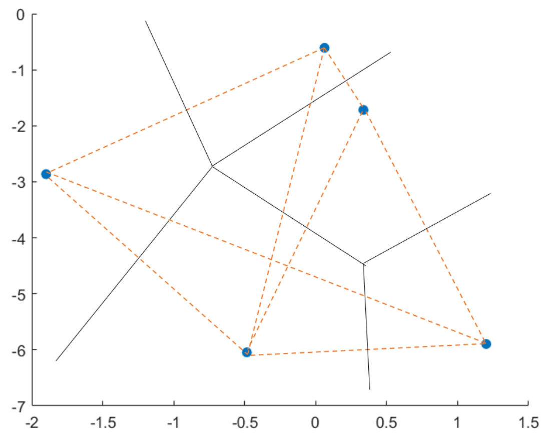


Figure 3: Decision regions are formed by combining the perpendicular bisectors (solid black) of the lines connecting the constellation points (dashed red).

This rule is called the minimum distance measure, or the nearest neighbor rule. With this minimum distance measure, it is easy to see that the boundaries are the perpendicular bisectors of the lines joining the signal constellation points (see Figure 3). Intuitively, this rule has the receiver select the symbol closest (in the Euclidean distance sense) to the received symbol. Mathematically, the choice to select the m-th symbol at the receiver is given by

$$D_m = \{m : \langle r, s_m \rangle + n_m > \langle r, s_{m'} \rangle + n_{m'}\}\,, \tag{7}$$

where $1 \leq m, m' \leq M$ and $m \neq m'$.

3.1 Two-state modulation

Consider a two-symbol communications scheme where the bit '1' is encoded using the symbol '+A', and the bit '0' is encoded using '-A', and transmitted over a noisy channel. Here, we will model the system and implement it using MATLAB.

To begin with, generate the data. Typically, when a large number of bits is transmitted, from a source modeled as random (such as speech or data), we assume an equal number of 1's and 0's are transmitted, making both symbols equally likely. Since we have two symbols, we can represent this in one dimension. We use the following MATLAB commands:

```
>> N = 1000; % Number of symbols to be transmitted
>> A = 1; % We can adjust this as needed. This is directly related to the transmit power
>> x = A*(2*(randn(N,1)>0)) - 1); % Generates +A and -A; both equally likely
```

Use a scatter plot to visualize these data points.

At the receiver, we model the additive channel noise as a zero-mean Gaussian noise. The received signal can be generated using MATLAB as:

```
>> var_n = 0.01; % Noise variance
>> n = randn(N,1)*sqrt(var_n); % Zero-mean Gaussian noise with variance var_n
>> r = x + n; % Received signal
```

Use a scatter plot to visualize these data points.

The receiver then makes a decision on each received signal. In this case, using the decision rule in (7) tells us that the received signal is called as '+A' if $r > 0$, and '-A' if $r < 0$.
What happens if $r = 0$?
Once again, visualize the received data on a scatter plot.

3.2 Error Characterization

Now that we have the received data, we need to evaluate how accurate the receiver is. We compare what was transmitted with what was received and count the number of symbols in errors. The rate of errors, that is, divide the number of errors by the number of transmitted bits, is also called the probability of error, P_e.

For the received data from Section 3.1, find the probability of error.

Note that as 'A' increases, the distance between the cluster centers and the decision boundary increases. This makes the system more resistant to noise. To verify this claim, set 'A' to be 2, and repeat the simulation. What is the new probability of error.

One thing to note here is that values of 'A' or the noise variance, individuslly do not affect performance; instead, the ratio,

$$\theta = \frac{A^2}{\sigma_n^2}, \tag{8}$$

defined as the signal-to-noise ratio, or SNR, is what drives the performance. A larger SNR provdes better performance, and *vice-versa*.

Run a simulation to compute probability of error vs SNR. Vary SNR from 10dB to 40dB, and plot the log of the probability of error vs the SNR in dB. Comment on your results.

Hints:

1. Increase the number of symbols until the curve becomes smooth.
2. To change the SNR, vary 'A'.
3. The dB value of a quantity X is given by:
$$X_{\mathrm{dB}} = 10\log_{10}(X). \tag{9}$$
4. Keep the SNR defined in dB amd compute to non-dB only as needed.
5. Use `semilogy` to plot the y-axis (probability of error) on a log scale. Leave the SNR in dB when you plot.

3.3 Multi-state Modulation

Extend the modulation scheme to two dimensions and four symbols, so that

$$\begin{aligned} s_1 &= (A, 0) \\ s_2 &= (0, A) \\ s_3 &= (-A, 0) \\ s_4 &= (0, -A) \end{aligned}$$

1. Derive and sketch the decision boundaries.

2. Run a simulation to compute probability of error vs SNR. Vary SNR from 10dB to 40dB, and plot the log of the probability of error vs the SNR in dB. Comment on your results. Use a large enough `N` to get a smooth curve.

Repeat for the modulation scheme:

$$\begin{aligned} s_1 &= (\frac{A}{\sqrt{2}}, \frac{A}{\sqrt{2}}) \\ s_2 &= (-\frac{A}{\sqrt{2}}, \frac{A}{\sqrt{2}}) \\ s_3 &= (-\frac{A}{\sqrt{2}}, -\frac{A}{\sqrt{2}}) \\ s_4 &= (\frac{A}{\sqrt{2}}, -\frac{A}{\sqrt{2}}) \end{aligned}$$

4 Reporting Requirements

Submit a project report that includes a brief description of what you did, and provide responses to all the prompts. Make sure the report is clearly organized with all figures numbered and labeled. The text should refer to each figure and describe the takeaways from each figure. Remember that the project grade depends on both the deliverables and the quality of the project report. Finally, at the end of the report, as an appendix, include all your MATLAB code.

Submit your report as a single .zip file containing:

1. Your report in .PDF format.

2. Your code

Name the file "`Project4_GroupX.zip`", where `X` is your group number.

Projects reports are due on November 13, 2020. Late reports will be accepted with a *20 point penalty* until November 20, 2020, and with a 30 point penalty until noon on November 24, 2020.