E1–244 **Detection and Estimation Theory**

Discussion: Mondays 4:00-5:00 PM
Due: 27 March 2015

Problem Set 5

Instructor: Rajesh Sundaresan *TA: None*

1. We solved the scalar estimation problem. Suppose instead that $\Lambda = \mathbb{R}^m$ and assume that the cost function is given by
$$C(a, \theta) = \sum_{i=1}^{m} (a_i - \theta_i)^2.$$
What is the optimal Bayes estimate? How would your answer change if you had a more general $C(a, \theta) = \sum_{i=1}^{m} C_i(a_i, \theta_i)$?

2. Coupled cost: Suppose that for the vector problem, we now have a coupled cost given by
$$C(a, \theta) = (a - \theta)^t A (a - \theta)$$
for some (symmetric) positive definite A. What is the optimal Bayes estimate for θ given Y? Are you surprised?

3. Suppose $Y_i = \Theta s_i + Z_i$, $i = 1, \ldots, n$, where $\Theta \sim N(\mu, v^2)$, $Z \sim N(0, K)$, and $s \in \mathbb{R}^n$ is a known signal. Find the MMSE estimate of Θ. Specialise to the case when $\Sigma = \sigma^2 I_n$, and $s = [1, 1, \ldots, 1]^T$. What is your answer when there are no samples ($n = 0$)? When $n \to \infty$? Interpret your answers.

4. Suppose that $Y = X + Z$ where $X = 1$ or -1 with equal probability and $Z \sim N(0, \sigma^2)$. Find the MMSE estimate for X given Y. Why might a real valued estimate for a binary valued random variable be useful?

5. In the above problem, what happens as $\sigma^2 \to +\infty$? As $\sigma^2 \to 0$?

6. Suppose $Y_i = \theta s_i + Z_i$, $i = 1, \ldots, n$, where θ is a scalar parameter, $Z \sim N(0, K)$, and $s \in \mathbb{R}^n$ is a known signal. Let the conditional distribution of $Y = [Y_1, \ldots, Y_n]^T$ given θ be denoted P_θ. Find a *scalar* sufficient statistic for the family $\{P_\theta, \theta \in \mathbb{R}\}$. (*Hint*: See Problem 3 above.)

7. See the programming exercise on equalisation techniques for multiple access channels in the next four pages.

1 Equalisers for Multiuser Detection

In this programming exercise, you will implement equalisation techniques for multiple access channels. We are particularly interested in the reception of one particular user in the presence of interference from other users and white Gaussian noise. You can follow the *Specific steps* outlined in one of the later sections. Some theory behind the LMS and the blind adaptive equalisers will follow a little later in the course.

2 Detection schemes

The detection schemes studied are:

- Matched Filter - This receiver directly matches to the transmitted pulse without taking into account multiple access interference.
- Minimum Mean Square Error (MMSE) Equaliser - This receiver minimises the mean squared error, where error is the difference between the equaliser output and the transmitted symbol (i.e., ± 1). To obtain this filter, we assume that the users' transmitted bits are independent and that their signatures are known. However, the transmissions of the various users may interfere with each other.
- Least Mean Square (LMS) Equaliser - This receiver operates without knowledge of the interfering users' signatures, but requires a training sequence.
- Blind Adaptive Equaliser - This receiver does not require a training sequence. All it needs is the signature waveform of the user of interest.

3 Model

We assume the following synchronous model where all users start transmission at the same time,

$$y = \sum_{j=1}^{L} A_j d_j s_j + n$$

where

- s_j : signature waveform of user j; belongs to $\{-1, 1\}^N$, normalised to have unit energy.
- A_j : amplitude of user j.
- d_j : transmitted symbol of user j; one of ± 1.
- n : white Gaussian noise vector composed of zero mean and unit variance components.
- y: received signal vector bearing signal, interference and noise.

4 Implementation

All receivers will be of the form

$$\hat{d}_1 = \text{sign}(c^t y)$$

i.e., we restrict our attention to linear Multi-User Detectors (MUD). Note that we are interested only in the detection of user 1's transmitted symbols.

You will implement the following receivers.

(a) **Matched filter** : $c = s_1$.

(b) **MMSE** : This receiver minimises $\mathbb{E}\left[(d_1 - c^t y)^2\right]$. The solution is given by

$$c = A_1 \left[\sum_{j=1}^{L} A_j^2 s_j s_j^t + \sigma^2 I_N\right]^{-1} s_1,$$

where I_N is the $N \times N$ identity matrix, and $\sigma^2 = 1$. Note that the MMSE is given by

$$\mathsf{MMSE} = 1 - A_1 c^t s_1.$$

(c) **LMS** : This is an iterative procedure that works with training across several symbols periods. At time k, let the error be denoted by

$$e(k) = d_1(k) - c^t(k-1)y(k).$$

Take $c(0) = s_1$. Then the adaptation rule is given by

$$c(k) = c(k-1) + 2\mu e(k)y(k).$$

where μ is a positive real parameter.

(d) **Blind adaptive equaliser** : We write

$$c = s_1 + x_1, \text{ where } ||s_1|| = 1 \text{ and } s_1^t x_1 = 0, \tag{1}$$

i.e., the equaliser has a signal component equal to the normalised matched filter and an orthogonal component. The adaptation rule adapts the orthogonal component by attempting to minimise $\mathbb{E}[(c^t y)^2]$ subject to the constraints in (1). The adaptation rule is given by

$$x(k) = x(k-1) - 2\mu Z(k)(y(k) - Z_{MF}(k)s_1),$$

where

$$\begin{aligned} Z(k) &= c^t(k-1)y(k), \\ Z_{MF}(k) &= s_1^t y(k), \\ c(k) &= s_1 + x(k). \end{aligned}$$

Note that this algorithm does not require any training sequence. Neither does it require knowledge of interfering users' signatures. All it needs is the desired user's signature.

5 Specific steps

1. Create a function `data = randBinary(numUsers, numSymbols)` whose output is a matrix of size `numUsers` $\times$ `numSymbols` that contains equilikely ± 1 data symbols.

2. Write a main program to test your `randBinary` function. The output data symbols should be printed on the screen. Test for several `numUsers` and several `numSymbols`.

3. Create a function `signatures = getSignatures(numUsers, sf,` ρ`)` whose output is such that the ith row is the signature sequence for user i. You may assume `sf` is a power of 2. The correlation between user 1 and user 2 ($s_1^t s_2$) should be as close to ρ as possible. All others users' signatures should be randomly generated. All signatures should have unit energy.

4. Append your main program to test `getSignatures`. Do this by printing out signatures for various values of `sf, numUsers,` ρ and verifying they are correct via prints on screen. For the rest of the session, use $\rho = 0.4$.

5. Create a function `rxIntf = generateInterference(signatures, data,` A`)` that is a vector of received signal (without noise) of size $1\times$ (`sf` $\cdot$ `numSymbols`), where A is a vector of gains. **No loops**.

6. Update your main program to test `generateInterference` via prints on screen. Verify for different values of A as well.

7. Create a function `rxSignal = getRxSignal(rxIntf, variance)` that models the received signal embedded in zero-mean white Gaussian noise of variance `variance`. The output vector should be `1` $\times$ `sf` $\cdot$ `numSymbols` in size.

8. Update your main program to test `getRxSignal` via prints on screen. Verify match with `rxIntf` for `variance` $= 0$. Do this for two users with $A = [1, 2]$. Then reshape `rxSignal` to a matrix of size `sf` $\times$ `numSymbols`. Each symbol period of received symbols should be in one column. For subsequent steps set `variance = 1`.

9. Create a function `errorRateMF = MF(signatures, rxSignal, data)`. The output should be the error rate for matched filter. Do the matched filter on each symbol, compute the number of errors, and compute the error rate. Note that `data` is used only for statistics generation. **No loops**.

10. Update your main program to test the above function. Set $A = [1, 0]$. Verify error rate for numSymbols = 10000 matches with theory (Q function evaluated at an appropriate point).

11. Create a function
    `[errorRateMMSE, estimatedMMSE, MMSE, cMMSE, MSETrajectory]`
    `= MMSE(signatures, rxSignal, data)`.
    The output should be the error rate, estimated MMSE, and true MMSE computed using formula, the MMSE filter, a trajectory of the MSE over time. (No loops. Use backslash or `mldivide` for computing $R^{-1}x$).

12. Update your main program to test the above function and plot the MSE trajectory. Set $A = [1, 0]$. The estimated MMSE should be close to the true MMSE for numSymbols = 10000. What should the MMSE filter be for this case? What should the error rate be for this case? For subsequent tests, set $A = [2, 1]$.

13. Create a function
    `[errorRateLMS, squaredErrorTrajectory, cLMS] = LMS(signatures, rxSignal, data,` μ`)`.
    The output should be the error rate averaged over time, the squared error at each time instant, and the converged LMS filter. Use the formulae provided. (A `for` loop is allowed for the time updates).

14. Update your main program to test the above function. Use `numSymbols = 500`. Take $\mu = 0.0001$. Get a plot of the quantity `squaredErrorTrajectory` versus time. Compare with MMSE filter's `MSETrajectory`. Compare `cLMS` and `cMMSE.`

15. Create a function `[errorRateBlind, energyTrajectory, cBlind] = blind(signatures, rxSignal, data, μ)`. The vector `energyTrajectory` should have kth component $Z(k)^2$. Use the formulae provided. (A `for` loop is allowed for the time updates).

16. Update your main program to test the above function. Use `numSymbols = 500`. Plot the `energyTrajectory` versus time. Compare the `cBlind` with `cMMSE`.

17. (Reading and homework questions) To which filter does the LMS algorithm converge? The blind adaptive algorithm?