EECS 16A Designing Information Devices and Systems I
Fall 2018 Homework 12

This homework is due November 16, 2018, at 23:59.
Self-grades are due November 20, 2018, at 23:59.

Submission Format
Your homework submission should consist of **two** files.

- `hw12.pdf`: A single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

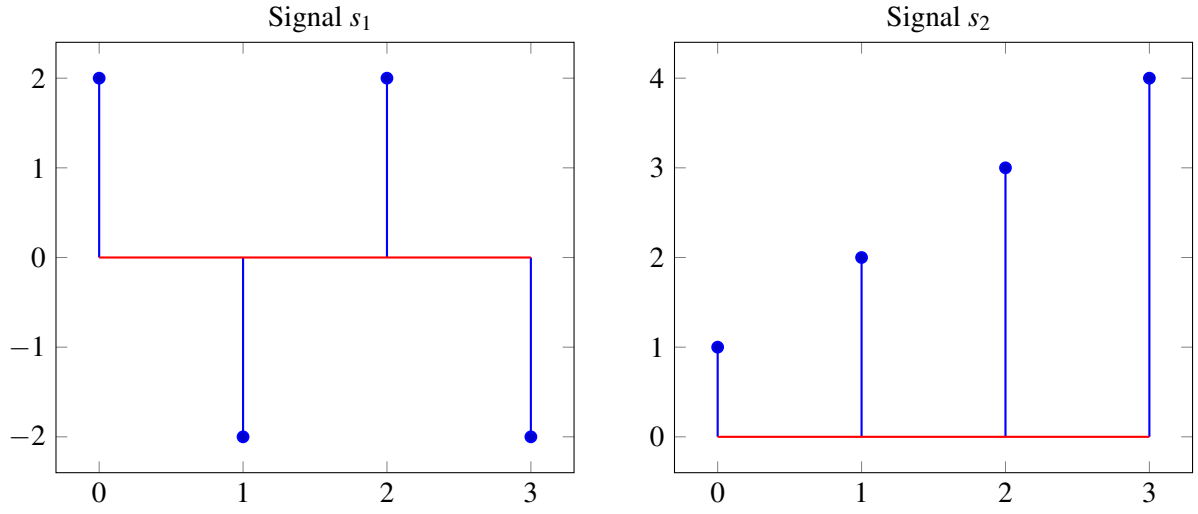
  If you do not attach a PDF of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible.

- `hw12.ipynb`: A single IPython notebook with all of your code in it.

  In order to receive credit for your IPython notebook, you must submit both a "printout" and the code itself.

Submit each file to its respective assignment on Gradescope.

1. Mechanical Correlation



(a) Assume that both of the above signals extend to $\pm\infty$, and are 0 everywhere outside of the region shown in the above graphs. Let $\vec{z}_1[n] = \vec{s}_2[n-1]$ and $\vec{z}_2[n] = \vec{s}_2[n+1]$. Plot $\vec{z}_1[n]$ and $\vec{z}_2[n]$.

(b) Now, assume each signal is periodic with period 4. Calculate and plot the linear **cross-correlation** for periodic signals for $\vec{s}_1$ and $\vec{s}_2$ (ie. $\text{corr}_N(\vec{s}_1, \vec{s}_2)$) and the linear **cross-correlation** for periodic signals for $\vec{s}_2$ and $\vec{s}_1$ (ie. $\text{corr}_N(\vec{s}_2, \vec{s}_1)$). Are they the same? How are they related?

(c) Again, assume that each signal is periodic with a period of 4 (one period shown). Calculate and plot the circular **auto-correlation** of each of the above signals. (ie. $circcorr(\vec{s}_1,\vec{s}_1)$ and $circcorr(\vec{s}_2,\vec{s}_2)$). The circular **auto-correlation** is found by computing the inner products of one period of the signal with all the possible shifts of one period of the same signal.

(d) Now, assume each signal is non-periodic and is 0 extending to $\pm$inf. Calculate and plot the linear **cross-correlation** of $\vec{s}_1$ and $\vec{s}_2$ (ie. $\text{corr}(\vec{s}_1,\vec{s}_2)$) and the linear **cross-correlation** of $\vec{s}_2$ and $\vec{s}_1$ (ie. $\text{corr}(\vec{s}_2,\vec{s}_1)$). Are they the same? How are they related?

(e) Using IPython, check your solution for part d using the Numpy command *np.correlate*. Please read this function's documentation. Which mode performs the linear cross-correlation?

(f) Let $corr_N(\vec{x},\vec{x})$ be the autocorrelation of an N-periodic signal $\vec{x}$. Prove that $corr_N(\vec{x},\vec{x})[0] \geq |corr_N(\vec{x},\vec{x})[m]|$ for all m. In other words, the autocorrelation peak (maximum value of autocorrelation) of any periodic signal always occurs at shift $m = 0$.

2. GPS Receivers

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. In this problem, we will understand how a receiver (e.g. your cellphone) can disambiguate signals from the different GPS satellites that are simultaneously received.

GPS satellites employ "spread-spectrum" technology (very similar to Code-division multiple access, CDMA, which is commonly used in cellphone transmissions) and a special coding scheme where each transmitter is assigned a code that serves as its "signature".

The GPS satellites use "Gold codes" which are 1023 bits long as their "signatures". You will use the ideas of correlation to figure out which of the satellites are transmitting.

The Gold codes have special properties:

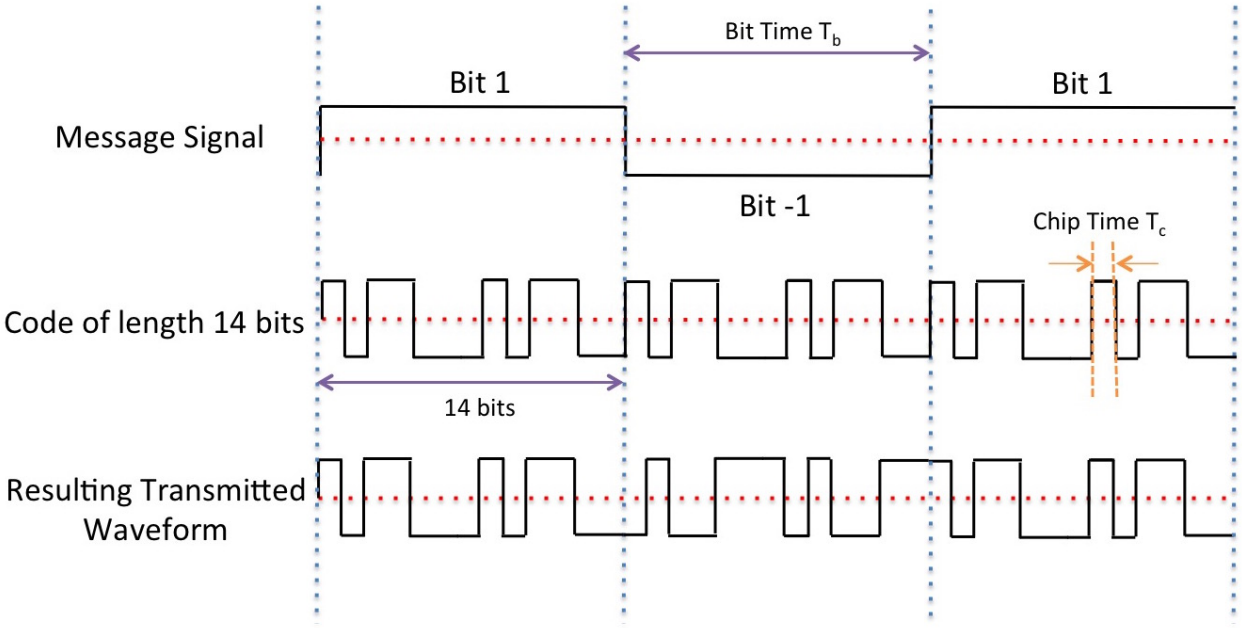
- The auto-correlation of a Gold code (correlation with itself) is very **high** at the 0^{th} shift and very **low** at all other shifts.
- The cross-correlation between different codes is very low.

These codes are generated using a linear feedback shift register (LFSR).You can read more about LFSR and CDMA if you are interested but for the problem you don't need to know how these work for the problem. The take-away is that the Gold codes are vectors of $+1$ and -1 values that are "almost orthogonal."

Sometimes transmitters (such as the GPS satellites) want to send additional "message bits" in addition to the signature they are transmitting. An example is depicted in the figure below. The *message signal* to be transmitted changes at a much slower timescale than the timescale of the signature code, which is very fast.

In the example figure, we are showing a message signal that is a stream of $+1$ or -1 values. The signature code is also a stream of $+1$ or -1 values of length 14 bits. The signature code is multiplied by the appropriate *message signal*, $+1$ or -1 values, to get the final transmitted waveform. Let T_b be the "bit time," i.e., the time for each message bit and T_c be the "chip time," which is the time for each new symbol of the code to be generated. In the figure, $T_b = 14T_c$.

For our problem, $T_b = 1023T_c$. (In reality, $T_b = 20 \times 1023 \times T_c$.)



For the purpose of this question we only consider 24 GPS satellites. Download the IPython notebook and the corresponding data files for the following questions:

(a) Auto-correlate the Gold code of satellite 10 with itself and plot it. Python has functions for this. What do you observe?

(b) Cross-correlate the Gold code of satellite 10 with satellite 13 and plot it. What do you observe?

(c) Now, consider a random signal, i.e. a signal that is not generated due to a specific code but is a random ± 1 sequence. Cross-correlate it with the Gold code of satellite 10. What do you observe? What does this mean about our ability to identify satellites?

(d) The signals received by a receiver include signals from the satellites as well as an additional noise term. This is often modeled as a Gaussian noise term. You don't need to understand Gaussians here, but we use them since they form a good model for how the transmitted signal might be perturbed (large perturbations are very unlikely, and small perturbations are more likely).

Use the Gaussian noise generator to generate a random vector of length 1023, and cross-correlate this with the Gold code of satellite 10. What do you observe?

For the next subparts of this problem, the signal is corrupted by Gaussian noise. Use the observation from this subpart for solving the rest of the question.

(e) Now, assume that signals from multiple satellites are added at the receiver, so the signatures of multiple different satellites are present in the code. In addition, noise might be added to the signal. What are the satellites present in `data1.npy`?

(f) Let's assume that you can hear only one satellite, Satellite A, at the location you are in (though this never happens in reality). Let's also assume that this satellite is transmitting a length 5 sequence of $+1$ and -1 after encoding it with the 1023 bit Gold code corresponding to Satellite A. Find out from `data2.npy` which satellite it is and what sequence of ± 1's it is transmitting.

(g) For the purpose of this problem, we'll assume that all the satellites transmit the same unique sequence of $+1$s and -1s. These are transmitted using the procedure described in the figure (called modulation, which we will learn more about soon.)

Signals from different transmitters arrive at the receiver with different propagation delays. Effectively, the signals from different satellites are superimposed on each other with different offsets at the start. This propagation delay is used to find out how far the satellite is from the receiver. To find out exactly what the offset due to propagation delay is, you want to figure out the starting point of the signal transmission, and you can do this by cross-correlating the signature codes with the received signal at different offsets. What do you expect to observe when you cross-correlate the signature for a particular satellite (say Satellite A) with the received signal at the offset corresponding to the propagation delay of Satellite A?

What satellites are you able to see in `data3.npy` and what are the relative delays assuming that the message signal that was being sent was exactly $\begin{bmatrix}1 & 1 & -1 & -1 & -1\end{bmatrix}$?

3. Retail Store Marketing

This problem describes a situation that online businesses face regularly. They can only observe the current purchases of a customer but would like to understand the general interests of the customer.

Intro The retail store EehEeh Sixteen would like to create an algorithm that can predict a customer's interests based only on the purchases made by the customer. Based on the interests, the store decides to give the customer a coupon (promotion) that is targeted to the right customer's interests.

Customer interests are described by a vector: $\vec{s_A} = \begin{bmatrix}\text{party-interest score}\\ \text{family-interest score}\\ \text{student-interest score}\\ \text{office-interest score}\end{bmatrix}$

The store would like to infer this vector for each customer.

(a) Assume we have the interests of a customer c in a vector $\vec{x_c} = \begin{bmatrix}c_{\text{party}}\\ c_{\text{family}}\\ c_{\text{student}}\\ c_{\text{office}}\end{bmatrix}$ and a set of promotions $A_1, A_2, \ldots, A_N$, with their attached vectors of scores $\vec{s_{A_1}}, \vec{s_{A_2}}, \ldots, \vec{s_{A_N}}$ (that are also customer interest vectors). We would like to select the promotion vector that is closest to the customer interest vector (minimizes the norm of the difference between the vectors). We would like to evaluate different measures of similarity — we want a function that outputs a higher value if the two vectors are closer to each other. The larger the value of the similarity function between $\vec{x_c}$ and $\vec{s}_{A_i}$ the better suited the promotion is for the customer.

Would $sim_1(\vec{x_c}, \vec{s_A}) = \|\vec{x_c} - \vec{s_A}\|$ be a good similarity measure? Why? What about $sim_2(\vec{x_c}, \vec{s_A}) = \left\langle \vec{x_c}, \frac{\vec{s_A}}{\|\vec{s_A}\|} \right\rangle$? Why?

(b) Unfortunately, the store does not get to observe each customer's interest vector. It only gets to observe the money the customer spends in four categories: food, movies, art, and books. The store needs to use this information to infer the vector $\vec{s_A}$ for each customer.

The EehEeh Sixteen research division conducted some studies that calculated the distribution of spending for people who are purely interested in only one category. For example, a person who is only interested in Party-spending will have the vector $x_c = [1,\ 0,\ 0,\ 0]^T$, and the spending of this person is given in the first line of Table: 1. Similarly, the remaining rows tell you how a person who is just interested in Family, Students, or Offices will spend.

	Spending Category			
Interest Category	**Food**	**Movies**	**Art**	**Books**
Party	40%	33%	22%	5%
Family	70%	10%	10%	10%
Student	20%	10%	15%	55%
Office	5%	2%	20%	73%

Table 1: The distribution of spending of people in each category.

We want to use this data to infer the interest vectors of customers given their spending, assuming that the spending of each customer is a linear combination of the spending of the "pure customers" (those with interest vectors $[1,\ 0,\ 0,\ 0]^T$, $[0,\ 1,\ 0,\ 0]^T$ etc). Suppose a customer spends $T_{\text{food}}\%$ on food, $T_{\text{movies}}\%$ on movies, $T_{\text{art}}\%$ on art, and $T_{\text{books}}\%$ on books.

Use the information in Table 1 to devise a system of linear equations so you can solve for the customer's preferences, x_c.

(c) Combine the results from the previous parts into a complete algorithm. The algorithm should take the raw spending of a customer, $M_{\text{food}}, M_{\text{movies}}, M_{\text{art}}, M_{\text{books}}$, and the promotion scores, $\vec{s_{A_1}}, \vec{s_{A_2}}, \ldots, \vec{s_{A_N}}$, as inputs. The algorithm's output should be the best promotion for that customer. For this part, use the second similarity metric from part a.

i. Since we no longer have access to the percentage spending of the customer, first normalize the spending subtotals to get spending percentages.

ii. Set up the system of linear equations in part c.

Algorithm 1 The EehEeh Sixteen promotions algorithm

procedure PROMOTION($M_{\text{food}}, M_{\text{movies}}, M_{\text{art}}, M_{\text{books}}, \vec{s_{A_1}}, \vec{s_{A_2}}, \ldots, \vec{s_{A_N}}$)
  $T_{\text{food}} =$
  $T_{\text{movies}} =$
  $T_{\text{art}} =$
  $T_{\text{books}} =$
  Setup and solve the system from part c
  Assign $\vec{x_c} = \begin{bmatrix} c_{\text{party}} \\ c_{\text{family}} \\ c_{\text{student}} \\ c_{\text{office}} \end{bmatrix}$
  Pick promotion A using similarity metric from part b.
  Print promotion A
end procedure

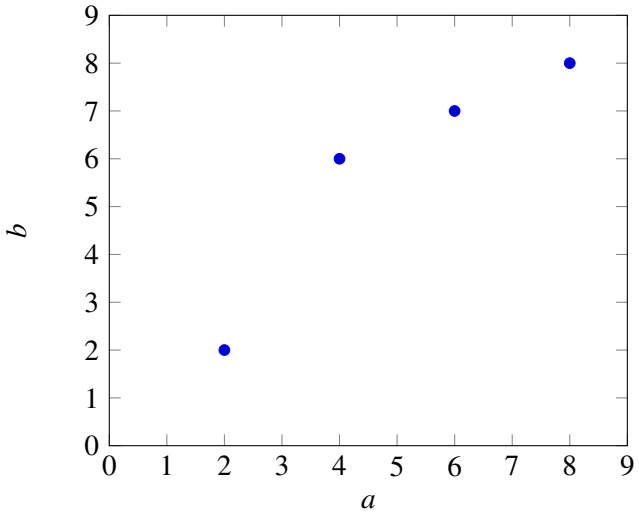
(d) Run the algorithm to figure out what promotion we should give to Jane Doe who spent \$6 on food, \$4 on movies, \$1 on art and \$5 on books. Use the values in Table 1 and assume there are 4 promotions, A_1,

A_2, A_3, and A_4, with associated score vectors $\vec{s_{A_1}} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$, $\vec{s_{A_2}} = \begin{bmatrix} \frac{2}{3} \\ -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{3} \end{bmatrix}$, $\vec{s_{A_3}} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{5}{2} \\ -\frac{1}{2} \end{bmatrix}$ and $\vec{s_{A_4}} = \begin{bmatrix} 0 \\ \frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix}$.

You may use IPython to run your algorithm. *Hint: Note that the preference vectors do not all have the same magnitude!*

(e) Will there ever be a customer for which the system devised in part (b) will yield no solutions or infinite solutions?

4. Mechanical: Linear Least Squares



a	2	4	6	8
b	2	6	7	8

(a) Consider the above data points. Find the linear model of the form

$$\vec{b} = \vec{a}x$$

that best fits the data, i.e. find the scalar value of x that minimizes

$$\left\| \begin{bmatrix} b_1 \\ \vdots \\ b_4 \end{bmatrix} - \begin{bmatrix} a_1 \\ \vdots \\ a_4 \end{bmatrix} x \right\|^2 = \|\vec{b} - \vec{a}x\|^2. \tag{1}$$

Do not use IPython for this calculation and show your work. Once you've computed x, compute the squared error between your model's prediction and the actual $\vec{b}$ values as shown in Equation 1. Plot the best fit line along with the data points to examine the quality of the fit. (It is okay if your plot of $\vec{b} = \vec{a}x$ is approximate.)

Reminder: $\hat{x} = (\vec{a}^T\vec{a})^{-1}\vec{a}^T\vec{b}$

(b) You will notice from your graph that you can get a better fit by adding a b-intercept. That is we can get a better fit for the data by assuming a linear model of the form

$$\vec{b} = x_1\vec{a} + x_2.$$

In order to do this, we need to augment our $\mathbf{A}$ matrix for the least squares calculation with a column of 1's (do you see why?), so that it has the form

$$\mathbf{A} = \begin{bmatrix} a_1 & 1 \\ \vdots & \vdots \\ a_4 & 1 \end{bmatrix}.$$

Find x_1 and x_2 that minimize

$$\left\| \begin{bmatrix} b_1 \\ \vdots \\ b_4 \end{bmatrix} - \begin{bmatrix} a_1 & 1 \\ \vdots & \vdots \\ a_4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\|^2. \tag{2}$$

Do not use IPython for this calculation and show your work.

Reminder: $\hat{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{b}$

Reminder: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

Compute the squared error between your model's prediction and the actual $\vec{b}$ values as shown in Equation 2. Plot your new linear model. Is it a better fit for the data?

(c) Let $\vec{\hat{x}}$ be the solution to a linear least squares problem.

$$\vec{\hat{x}} = \underset{\vec{x}}{\operatorname{argmin}} \left\| \vec{b} - \mathbf{A}\vec{x} \right\|^2$$

Show that the error vector $\vec{b} - \mathbf{A}\vec{\hat{x}}$ is orthogonal to the columns of $\mathbf{A}$ by direct manipulation (i.e. plug the formula for the linear least squares estimate into the error vector and then check if $\mathbf{A}^T$ times the vector is the zero vector.)

5. Write Your Own Question And Provide a Thorough Solution.

Do you have a cool idea that could make a great homework problem? Is there an application you wished you could see discussed in class/homework/discussion?

Writing your own problems is a very important way to really learn material. Having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams. You don't have to write a great problem on every attempt — each of the homework problems you seen have been iteratively improved over time. If you keep trying to write problems, you will get better at it.

6. Homework Process and Study Group

Who else did you work with on this homework? List names and student ID's. (In case of homework party, you can also just describe the group.) How did you work on this homework?