EECS 16A Designing Information Devices and Systems I
Fall 2016 Babak Ayazifar, Vladimir Stojanovic Homework 12

This homework is due November 22, 2016, at 1 P.M.

Recommended Reading: Gilbert Strang, *Introduction to Linear Algebra*, Chapter 4

1. **Homework process and study group**

   Who else did you work with on this homework? List names and student ID's. (In case of hw party, you can also just describe the group.) How did you work on this homework?
   Working in groups of 3-5 will earn credit for your participation grade.

2. **Mechanical Gram-Schmidt**

   (a) Use Gram-Schmidt to find a matrix U whose columns form an orthonormal basis for the column space of V.

   $$V = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

   and show that you get the same resulting vector when you project $w = [\, 1,\ -1,\ 0,\ -1,\ 0 \,]^T$ onto V and onto U, i.e. show that

   $$V(V^TV)^{-1}V^Tw = U(U^TU)^{-1}U^Tw \tag{2}$$

   (b) Use Gram-Schmidt to find a matrix U whose columns form an orthonormal basis for the column space of V.

   $$V = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & -1 & 1 \end{bmatrix} \tag{3}$$

   and show that you get the same resulting vector when you project $w = [\, 1,\ -1,\ 0,\ -1,\ 0 \,]^T$ onto V and onto U.

   (c) Compute the QR decomposition of the above matrices, that is, find an othogonal matrix **Q** and an upper-triangular matrix **R**, such that $\mathbf{V} = \mathbf{QR}$.

3. **The Framingham Risk Score**

   For most of the parts of this problem, your work will be done in the appropriate section of the ipython notebook.

   In Homework 1, we did a problem where we calculated the parameters of the Framingham risk score for predicting cardiovascular disease (CVD). In this problem, we will revisit the parameters of the Framingham risk score in a more realistic setting using the more sophisticated optimization tool of linear least squares. In the problem in Homework 1, we determined four parameters of Framingham risk score from the data from four patients – this amounts to solving four equations with four unknowns. This makes sense if we knew the correct parameters originally but then forgot them. Suppose, however, that we were trying to come up with the correct parameters for the Framingham risk score in the first place. How would we do it?

As a review, the Framingham risk score estimates the 10-year cardiovascular disease (CVD) risk of an individual. There are multiple factors (predictors) that weigh in the calculation of the score. In Homework 1, we simplified the score to only use four factors. Here we will look at more complex version of the score that takes into account six factors including age, total cholesterol, level of high-density lipoprotein (HDL) cholesterol, systolic blood pressure (SBP), whether or not the individual smokes, and whether or not the individual is diabetic.

Scores like this are determined empirically after tracking the characteristics of many medical patients. Once we have data from hundreds or thousands of test subjects, we want to find the parameters that best model the data we are seeing so that we can use our score to predict the probability of heart disease for a new patient. Of course there will be some variability in the probability of heart to disease for each individual but we want to design the parameters of our score so that it predicts their risk as closely as possible.

Linear least squares is a powerful tool for fitting these kind of models to minimize the error between the observed risk of heart disease for each individual and the predicted risk from the model. Linear least squares can even be a powerful tool in many cases when we expect our model to be nonlinear in the data. As long as we can transform the problem so that the model is **linear in the parameters** then we can use linear least squares. For example in the Framingham case, we have reason to believe (from medical modeling considerations) that the probability of the individual suffering from CVD in the next 10 years has the form

$$p = 1 - 0.95^{e^{(R-26.1931)}} \tag{4}$$

where the score R is calculated based on the values of age, total cholesterol (TC), HDL cholesterol, systolic blood pressure (SBP), whether or not the patient is diabetic (DIA), and whether or not the patient smokes (SMK) as follows

$$\begin{aligned} R = & x_1 \cdot \ln\Big(\text{AGE (years)}\Big) + x_2 \cdot \ln\Big(\text{TC (mg/dL)}\Big) + \\ & x_3 \cdot \ln\Big(\text{HDL (mg/dL)}\Big) + x_4 \cdot \ln\Big(\text{SBP (mm Hg)}\Big) + \\ & x_5 \cdot \Big(\text{DIA (binary)}\Big) + x_6 \cdot \Big(\text{SMK (binary)}\Big) \end{aligned} \tag{5}$$

DIA and SMK are binary variables that indicate whether or not the subject has diabetes and smokes respectively whereas AGE, TC, HDL, and SBP are all numeric values. Note also that AGE, TC, HDL, and SBP are passed through the natural log function $\ln(\cdot)$ where as DIA and SMK are not. For patient k, we will represent the probability as p^k and the score as R^k.

(a) We want to transform the probabilities and the input data (AGE, TC, HDL, SBP, DIA, SMK) for patient k into the form

$$b^k = x_1 A_1^k + x_2 A_2^k + x_3 A_3^k + x_4 A_4^k + x_5 A_5^k + x_6 A_6^k \tag{6}$$

in order to solve for the parameters $\vec{x} = [x_1,\, x_2,\, x_3,\, x_4,\, x_5,\, x_6]^T$. How can we transform the probabilities and the input data to express Equation (4) in the form of Equation (6), i.e. express b^k, A_1^k, A_2^k, A_3^k, A_4^k, A_5^k and A_6^k be in terms of p^k, AGE^k, TC^k, HDL^k, SBP^k, DIA^k, SMK^k. In the ipython notebook, load in the data file `CVDdata.mat` and apply these transformations to the appropriate variables.

Credit: The data was obtained from the Center for Disease Control and Prevention's (CDC) National Health and Nutrition Examination Survey (NHANES) dataset (October 2015). http://www.cdc.gov/nchs/nhanes.htm

(b) Now that we have transformed the problem into a linear problem, we want to use linear least squares to estimate the parameters $\vec{x}$. In order to do this we set up a system of equations in matrix form

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

where A is a tall matrix and $\vec{b}$ is a tall vector. What form should $\vec{b}$ and A have in terms of b^k, A_1^k, A_2^k, A_3^k, A_4^k, A_5^k, A_6^k? The data we loaded in python is for 91 patients. Construct $\vec{b}$ and A using the loaded data. What are the dimensions of $\vec{b}$ and A?

(c) We want to choose our estimate $\vec{\hat{x}}$ to minimize $||\vec{b} - A\vec{x}||^2$. Use the linear least squares formula to find the best fit parameters $\vec{\hat{x}}$. What is the $\vec{\hat{x}}$ that you found?

(d) Now that we've found the best fit parameters $\vec{\hat{x}}$, write an expression for $\vec{\hat{b}}$, our model's prediction of the values of $\vec{b}$ given the data A and our estimate of the parameters $\vec{\hat{x}}$ and compute the squared error $||\vec{b} - \vec{\hat{b}}||^2$. What is the squared error that you computed?

(e) Since this problem has many parameters it is difficult to visualize what is going on. One thing we can do to get a feel for the data and check that our fit is good is to plot it in a lower dimensional subspace. For example, we could plot b by A_1 or A_2 or A_3 individually. (A_1 is the vector of A_1^k's for all patients. It is the first column of A. Similarly for A_2 and A_3.) In your ipython notebook, plot b by A_1, b by A_2, and b by A_3 individually using the plotting option `'ob'` to plot the data as blue dots. For each plot you should see a blue point cloud. What is actually happening here is that we're projecting the data onto the A_1-b plane, the A_2-b plane, and the A_3-b plane respectively. Now plot your models prediction $\hat{b}$ by A_1, A_2, and A_3 on the appropriate plots using the plotting option `'or'` to plot the predictions as red dots (refer to the ipython notebook for reference code). Does it look like your model's fit is good?

(f) To better visualize the linearity of the model, we will calculate the risk as a function of A_2 alone and plot it. The rest of the predictors will be fixed in this part. We will use the following values for the other parameters. Age=55 years, HDL=25 mg/dL, SBP=220 mm Hg, DIA=1 and SMK=1. In the IPython notebook, we have generated a block of code that you need to complete to make the calculation of predicted b values from the above parameters. Fill the code and plot the estimated b values vs A_2 values. Is the plot linear?

Hint: don't forget to apply the appropriate transformation to the different parameters.

(g) Try changing the parameters $\vec{\hat{x}}$ slightly and re-plotting. Does it look like the fit is getting better or worse? Is the squared error increasing or decreasing?

(h) (BONUS) Transform b and $\hat{b}$ back into the form of p and transform A_1, A_2, A_3 back into the form of AGE, TC, and HDL and re-plot. What do you see?

(i) (BONUS) Use the values for b from part (f) to calculate p as a function of TC. Plot the curve p vs TC. Is the plot linear? What does this plot portray?

Note: Some of the values in the algorithm were modified from the original study values.

4. Finding Signals in Noise

Disclaimer: This problem looks long. But almost all of the parts only involve running provided IPython code, and commenting on the output. The last part is open-ended, but can be done by re-using code from the previous parts. It is important to understand the previous parts in order to tackle the last part.

In this problem, we will explore how to use correlation and least-squares to find signals, even in the presence of noise and other interfering signals.

(a) Suppose there is a transmitter sending a known signal s_1, which is periodic of length $N = 1000$. Say $\vec{s}_1$ is chosen to be a random $\{+1, -1\}$ vector (for example, by tossing a coin N times and replacing every heads with $+1$ and every tails with -1). For convenience, let us also normalize s_1 to norm 1. That is, the vector $\vec{s}_1$ looks like:

$$\vec{s}_1 = \frac{1}{\sqrt{n}} \begin{bmatrix} +1 & -1 & -1 & +1 & -1 & \dots \end{bmatrix}$$

(Where the ± 1 entries are chosen by random coin toss).

We claim that such a vector $\vec{s}_1$ is "approximately orthogonal" to circular-shifts of itself. That is, if $\vec{s}_1^{(j)}$ denotes circularly-shifting s_1 by j, then for all $j \neq 0$:

$$\left| \left\langle \vec{s}_1, \vec{s}_1^{(j)} \right\rangle \right| \lessapprox \varepsilon$$

for some small epsilon.

Run the provided IPython code to generate a random $\vec{s}_1$, and plot its autocorrelation (all inner-products with shifted versions of itself). Run this a few times, for different random vectors $\vec{s}_1$. Around how small are the largest inner-products $\left\langle \vec{s}_1, \vec{s}_1^{(j)} \right\rangle$, $j \neq 0$?

Recall that we normalized such that $\langle \vec{s}_1, \vec{s}_1 \rangle = 1$.

(b) Suppose we received a signal y, which is s_1 delayed by an unknown amount. That is,

$$\vec{y} = \vec{s}_1^{(j)}$$

for some unknown shift j. To find the delay, we can choose the shift j with the highest inner-product $\left\langle \vec{s}_1^{(j)}, \vec{y} \right\rangle$.

Run the provided IPython code to plot the cross-correlation between $\vec{y}$ and $\vec{s}_1$ (ie, all the shifted inner-products). Can you identify the delay? Briefly comment on why this works, using the findings from the previous part. *(Hint: What does $\left\langle \vec{s}_1^{(k)}, \vec{y} \right\rangle$ look like, for $k = j$? For $k \neq j$?)*

(c) Now suppose we received a slightly noisy signal:

$$\vec{y} = \vec{s}_1^{(j)} + 0.1\vec{n}$$

where the "noise" source $\vec{n}$ is chosen to be a random normalized vector, just like s_1.

Run the provided IPython code to compute $\langle \vec{s}_1, \vec{n} \rangle$. Run this a few times, for different random choices of s_1, n. Around how small is $| \langle \vec{s}_1, \vec{n} \rangle |$? How does this compare to $\left\langle \vec{s}_1, \vec{s}_1^{(j)} \right\rangle$, from your answer in part (a)?

(d) Can we identify the delay from this noisy reception? In this case, we do not know the noise $\vec{n}$, and we do not know the delay j. (But, as before, we know the signal $\vec{s}_1$ being transmitted).

Run the provided IPython code to plot the cross-correlation between $\vec{y}$ and $\vec{s}_1$. Briefly comment on why this works to find the delay, using the findings from the previous part.

(e) What if the noise was higher? For example:

$$\vec{y} = \vec{s}_1^{(j)} + \vec{n}$$

Does cross-correlation still work to find the delay? (Use the provided IPython notebook).

What about very high noise?

$$\vec{y} = \vec{s}_1^{(j)} + 10\vec{n}$$

Does cross-correlation still work to find the delay? If not, can you explain why? (use findings from previous parts).

(f) Now suppose there are two transmitters, sending known signals $\vec{s}_1$ and $\vec{s}_2$ at two unknown delays. That is, we receive

$$\vec{y} = \vec{s}_1^{(j)} + \vec{s}_2^{(k)}$$

for unknown shifts j,k. Both signals $\vec{s}_1$ and $\vec{s}_2$ are chosen as random normalized vectors, as before.

We can try to find the first signal delay by cross-correlating $\vec{y}$ with $\vec{s}_1$ (as in the previous parts). Similarly, we can try to find the second signal delay by cross-correlating $\vec{y}$ with $\vec{s}_2$. Run the provided IPython code to estimate the delays j,k. Does this method work to find both delays? Briefly comment on why or why not.

(g) Now, suppose the second transmitter is very weak, so we receive:

$$\vec{y} = \vec{s}_1^{(j)} + 0.1\vec{s}_2^{(k)}$$

Does the method of the previous part work reliably to find signal s_1? What about s_2? (Run the provided code a few times, to test for different choices of random signals). Briefly justify why or why not.

(Hint: $\vec{s}_1$ looks like "noise" when we are trying to find $\vec{s}_2$. Based on the previous parts, would you expect to be able to find $\vec{s}_2$ under such high noise?)

(h) To address the problem of the previous part, suppose we use the following strategy: First, cross-correlate to find the delay j of the strongest signal (say, $\vec{s}_1$). Then, subtract this out from the received $\vec{y}$, to get a new signal $\vec{y}' = \vec{y} - \vec{s}_1^{(j)}$. Then cross-correlate to find the second signal in $\vec{y}'$.

Run the provided IPython code to test this strategy for the setup of the previous part (with a strong and weak transmitter). Does it work? Briefly comment on why or why not.

(i) Finally, suppose the amplitudes of the sent signals are also unknown. That is:

$$\vec{y} = \alpha_1\vec{s}_1^{(j)} + \alpha_2\vec{s}_2^{(k)}$$

for unknown amplitudes $\alpha_1 > 0$, $\alpha_2 > 0$, and unknown shifts j,k.

Can we use inner-products (cross-correlation) to find the amplitudes as well? For example, suppose we find the correct shift j via cross-correlation. Then, briefly comment on why the following holds:

$$\left\langle \vec{s}_1^{(j)}, \vec{y} \right\rangle \approx \alpha_1$$

Run the provided IPython notebook to try this method of estimating the coefficients α_1, α_2. Roughly how close are the estimates to the actual amplitudes? (Run the code a few times, to test different choices of random signals).

(j) Repeat the above for when there is some additional noise as well:

$$\vec{y} = \alpha_1\vec{s}_1^{(j)} + \alpha_2\vec{s}_2^{(k)} + 0.1\vec{n}$$

Roughly how close are the estimates to the actual amplitudes? (Run the code a few times, to test different choices of random signals).

(k) Let us try to improve the coefficient estimates. Once we have identified the correct shifts j,k, we can setup a Least-Squares problem to find the "best" amplitudes α_1, α_2 such that

$$\vec{y} \approx \alpha_1\vec{s}_1^{(j)} + \alpha_2\vec{s}_2^{(k)}$$

Set up a Least-Squares problem in the form

$$\min ||A\vec{x} - \vec{b}||$$

to estimate α_1, α_2. What is the matrix A and vector b?

(Hint: A will have 1000 *rows and* 2 *columns)*

Use the provided IPython notebook to try this method of estimating the coefficients α_1, α_2. Roughly how close are the estimates to the actual amplitudes? Did this improve on the strategy from the previous part?

(l) Now try to use these tools on your own. Suppose there are 3 transmitters, transmitting known signals $\vec{s}_1, \vec{s}_2, \vec{s}_3$. These signals were chosen as random normalized vectors (as above). You received a combination of these signals at unknown delays and unknown amplitudes, and corrupted by some unknown noise:

$$\vec{y} = \alpha_1 \vec{s}_1^{(j)} + \alpha_2 \vec{s}_2^{(k)} + \alpha_3 \vec{s}_3^{(l)} + 0.005\vec{n}$$

(Where $\vec{n}$ was also chosen as a random normalized vector as above).

You are provided with the known signals $\vec{s}_1, \vec{s}_2, \vec{s}_3$ and the received signal $\vec{y}$ in the IPython notebook. **Try to find the unknown delays** j,k,l. *(Hint: Some of the signals may be weak).*

There is a test function provided that will try to “decode” your candidate shifts j,k,l into a message. You should recognize the message.

Describe your procedure, and include your entire workflow (any plots you generated, etc) in your IPython solutions.

5. Deconstructing Trolls

(a) Consider two vectors that are orthogonal to each other, $\vec{a}$ and $\vec{b}$. Suppose we scale both vectors by some scalar c. Are they still orthogonal? Now suppose we scale them by two different scalars, c_1 and c_2. Are they still orthogonal?

(b) Now, let’s construct two new vectors $\vec{x}_1$ and $\vec{x}_2$ that are both linear combinations of $\vec{a}$ and $\vec{b}$. In general, are $\vec{x}_1$ and $\vec{x}_2$ orthogonal? If yes, provide some justification, if not, provide a counter example.

(c) Suppose now we have a matrix **M** whose columns are $\vec{x}_1$ and $\vec{x}_2$. We find the QR factorization of **M**. What is one possible Q matrix? Recall that the Q matrix is a set of orthogonal vectors that span the column space of the matrix M.

(d) Now let’s revisit the troll problem we saw in Homework 1. Recall there were two speakers and two microphones. The microphones recorded sound that was a linear combination of the two speakers. More specifically, Let the sound of the two speakers be represented by length n vectors $\vec{a}$ and $\vec{b}$ and the sound recorded by the microphones be represented by length n vectors $\vec{m}_1$ and $\vec{m}_2$. We know $\vec{a}$ and $\vec{b}$ are orthogonal, and that $\vec{m}_1$ and $\vec{m}_2$ are linear combinations of $\vec{a}$ and $\vec{b}$. Given just $\vec{m}_1$ and $\vec{m}_2$, devise a way to find $\vec{a}$ and $\vec{b}$. *(HINT: Use QR decomposition)*

(e) Implement your method in the provided notebook and commment on the results.

6. Your Own Problem Write your own problem related to this week’s material and solve it. You may still work in groups to brainstorm problems, but each student should submit a unique problem. What is the problem? How to formulate it? How to solve it? What is the solution?