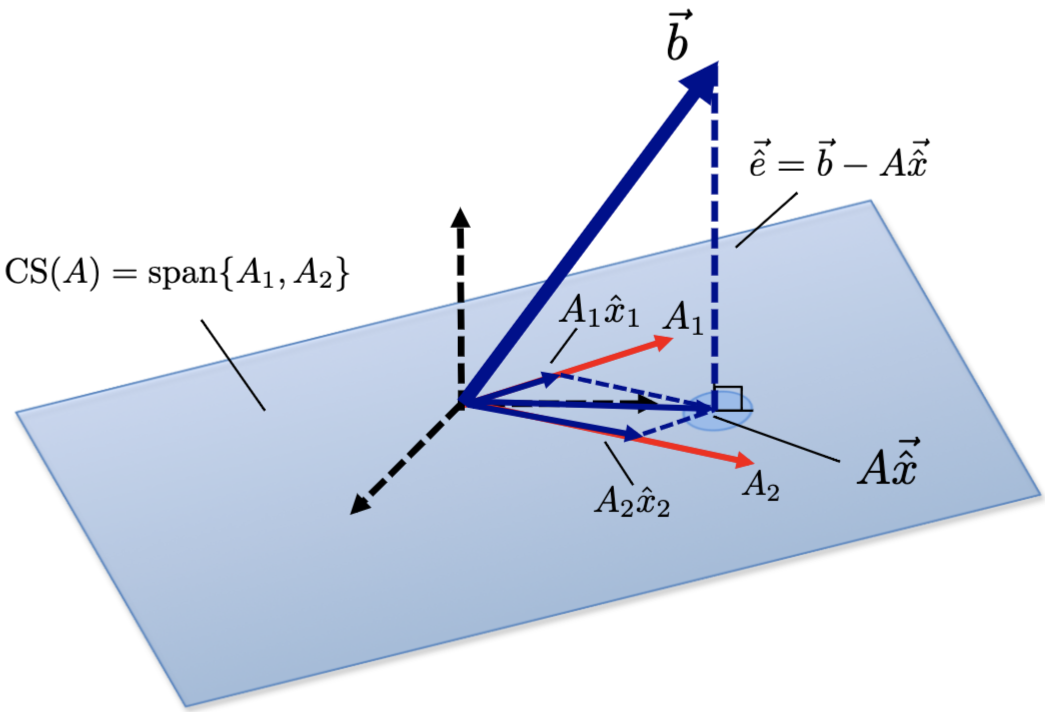
CSM 16A	Designing Information Systems and Devices

Week 13 Worksheet

Term: **Spring 2020** *Name:*

Problem 1: Least Squares and Orthogonal Projection



1. Consider that you have some equations of the form $\mathbf{A}\vec{x} = \vec{b}$, however, that there is no solution $\vec{x}$ that solves the equations. What does this tell us about $\vec{b}$ with respect to $\mathbf{A}_1, \mathbf{A}_2$ (the columns of $\mathbf{A}$)?

2. We know that there is no $\vec{x}$ that satisfies the equations exactly, but we still want to solve the equations to get a solution as close as possible.

   Let's say you had 3 choices, $\vec{x}_i, \vec{x}_j, \vec{x}_k$ (these are not drawn on the image). What could you compute in order to determine which of these would be the best choice instead of $\vec{x}$

3. Suppose that the real vector $\vec{b} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, and you have two vectors $\vec{x}_1, \vec{x}_2$ that are close to $\vec{b}$, which result in possible $\vec{b}_1 = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix}$ and another $\vec{b}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. How do we know which one is *closer* to $\vec{b}$? What if we define *the distance between two vectors* as the sum of the components of the difference of the vectors.

4. What is a different approach to solve the issue discussed above?
   Hint: Consider the euclidean distance between 2 points p_1 and p_2.

5. Using this definition, how close is $\vec{b}_1$ to $\vec{b}$? How close is $\vec{b}_2$ to $\vec{b}$?

6. More generally, we actually don't have a choice of just two or three $\vec{x}$s to pick to get as close to $\vec{b}$ as possible. We have an infinite number of choices. How can we tell which one is the best? Look at the image given at the top of the question, and decide something about $\vec{\hat{x}}$ and $\vec{\hat{e}}$.

7. Let's begin by recalling three facts. Recall that if we want to minimize some quantity squared, it is enough to minimize the quantity itself. Also, recall that given a point and a plane, the shortest line that one can possibly get starting at the point and ending at the plane, is a perpendicular line from the point to the plane. Finally, recall that if a vector $\vec{v}$ is orthogonal to another vector u, their inner product $\langle \vec{v}, \vec{u} \rangle = \vec{v}^T \vec{u} = 0$. Using this information, come up with a method to minimize the norm-squared of $\vec{\hat{e}}$.

Problem 2: Least Squares

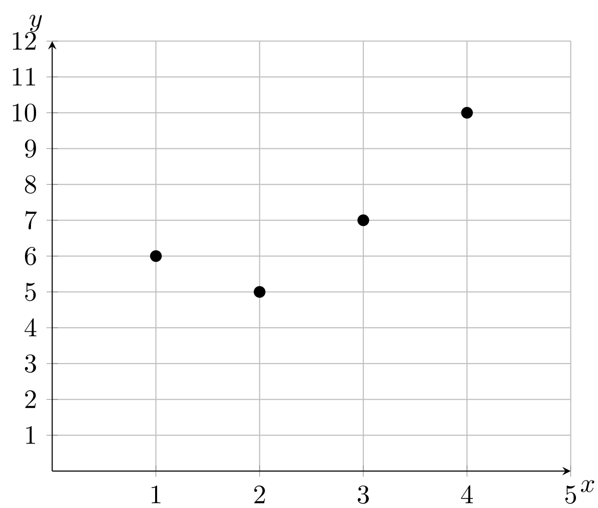
Consider the basic problem of finding some $\vec{x}$ such that $\mathbf{A}\vec{x} = \vec{b}$. If we have an equal number of equations as unknowns, then we are pretty happy.

In many cases, however, we cannot find a solution, as the set of equations that are described by $\mathbf{A}$ and $\vec{b}$ are overdetermined (i.e. there are more equations than there are unknowns).

In general, least squares involves finding the best approximate solution to these overdetermined systems.

Looking at this graphically, we can think of the problem of data-fitting. That is to say that we have many samples, and we want to draw a straight line that goes as close to each point as possible.

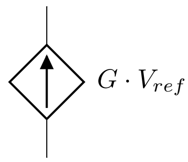
1. Looking at the plot below, setup a system of linear equations describing a line going through each point.



2. Put this system of linear equations into matrix-vector form.
3. Given that $(\mathbf{A}^T\mathbf{A})^{-1} = \begin{bmatrix} 0.2 & -0.5 \\ -0.5 & 1.5 \end{bmatrix}$, use the linear least squares technique you learned earlier on this overdetermined system to solve for a line of best fit
4. Plot this line in the plot above.
5. C.F. Gauss (the 19th century mathematician of Gaussian elimination fame) used least squares to model the orbits of various celestial bodies around the sun. From Kepler's laws of planetary motion, Gauss constructed the following equation:
$$\alpha x^2 + \beta xy + \gamma y^2 + \delta x + \epsilon y = 1$$
where x and y describe the position of the object in the 2D plane of its orbit around the sun, and the Greek letters are unknown coefficients. A scientist named Piazzi observed the orbit of the dwarf planet Ceres over the period of a month, producing 19 data points of its xy-coordinates. Given this data, how might Gauss have used least squares to recover appropriate values of the coefficients alpha, beta, gamma, delta, epsilon to model the orbit of Ceres?

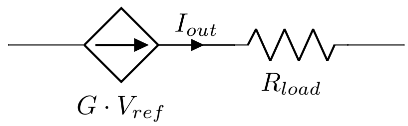
Problem 3: Circuit Design

A Voltage Controlled Current Source looks like this:

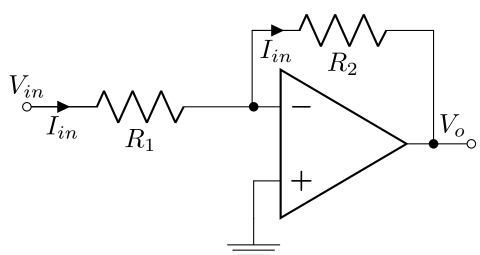


A VCCS is a Dependent Current Source that's controlled by a voltage V_{ref}, and produces a current based on that V_{ref}, which will follow the equation $I_{out} = G \cdot V_{ref}$, for some constant G.

It can be connected to any load resistor (it has a resistance of R_{load}), and guarantees that the current $G \cdot Vref$ will flow through that resistor.



1. In order to create a VCCS, we'll need some way to turn an input voltage into a current. What's the simplest way we can accomplish this?
   Hint: What's a circuit element that relates voltage and current?
2. We'll also need some way to isolate the input voltage from the previous parts of the circuit that produces our input voltage. In other words, if the input voltage to our VCCS has some resistors or other components connected to it, we don't want that to affect the relation between V_{ref} and I_{out} in our VCCS. What design component can we use to do this?
3. Now we have V_{ref} converted to I_{out} using the formula $G \cdot V_{ref}$. What is G in terms of R_{ref}?
4. Are we done? Are there any problems with the current that we're producing?
   (Hint: What happens when we place our R_{load} at I_{out}?)
5. We can keep the current at the value we want it to be at by using one of the properties of an inverting op amp: the fact that no matter what, the current flowing through the $V-$ to V_{out} branch is equal to the input current to $V-$ (This follows from the fact that no current flows into $V-$).

   Where can we place our previous circuit and R_{load} to take advantage of this? Draw the entire circuit configuration for your VCCS.

Problem 4: (Challenging) Disease Diagnosis

Consider a set of patients. Patient i can be represented by an attribute vector $\vec{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$ as well as a known label $y^{(i)} \in \{-1, +1\}$ indicating whether they have a disease D. We want to design a simple classifier that can use the information from the data we have in order to predict whether a patient with attributes $x_1^{(i)}$ and $x_2^{(i)}$ and unknown diagnosis status has the disease. To do this, we will use our knowledge of least squares linear regression. We would like to design a linear function

$$f\left(\vec{x}^{(i)}\right) = \vec{w}^T \vec{x}^{(i)}$$

that takes in a vector $\vec{x}^{(i)}$ for a patient i and computes $y^{(i)} = \text{sign}\left(f(\vec{x}^{(i)})\right)$ to predict whether the patient has the disease.

Here, $\vec{w}$ represents the vecotr containing all the weights (coefficients) for each of the attribute entries in the vector $\vec{x}^{(i)}$.

Note: $\text{sign}(x) = \begin{Bmatrix} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{Bmatrix}$

1. In order to make our classifier as accurate as possible, we've found the following 4 error functions that we can choose from to minimize. choose **one that best suits our model** and **explain why the other three don't work as well** in this scenario.
   - Linear Error: $\vec{e} = \vec{y}^{(i)} - f(\vec{x}^{(i)})$
   - Absolute Error: $\vec{e} = |\vec{y}^{(i)} - f(\vec{x}^{(i)})|$
   - Squared Error: $\vec{e} = \left(\vec{y}^{(i)} - f(\vec{x}^{(i)})\right)^2$
   - Sign Error: $\vec{e} = \begin{cases} +1, \text{if } \vec{y}^{(i)} = f(\vec{x}^{(i)}) \\ -1, \text{if } \vec{y}^{(i)} \neq f(\vec{x}^{(i)}) \end{cases}$
2. Suppose we have a brand new set of data points as shown in the table below along with a column representing the actual diagnosis results: (+1) for positive diagnosis and (-1) for negative diagnosis. Given that we've found the weight vector $\vec{w}$ of our linear function $f(\vec{x}^{(i)})$ to be $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$, predict whether each patient has the disease by filling in the table with the prediction **Yes, No,** or **Inconclusive**. What is the accuracy of our linear classifier?

Patient	Attribute x_1	Attribute x_2	Diagnosis Result	Prediction from the Classifier
1	1	0	(+1)	
2	-6	1	(-1)	
3	-5	-5	(+1)	
4	9	-6	(-1)	

   Note: As a refresher, here is how we will make the final prediction based on the value of our linear function:
   - When $f(\vec{x}^{(i)}) > 0$, the classifier outputs **Yes**.
   - When $f(\vec{x}^{(i)}) < 0$, the classifier outputs **No**.
   - When $f(\vec{x}^{(i)}) = 0$, the classifier outputs **Inconclusive**.
3. Applying the same model, we have found a different linear classifier $f(\vec{x}^{(j)}) = \vec{w}^T \vec{x}^{(j)}$ for another disease T with 2 attributes from each patient. Surprisingly, the disease has an astonishingly high accuracy of 95 % on all the patients we have trained our classifier on so far.

(a) In order to further validate our model, we obtain a new batch of data on 50 more patients along with their diagnosis results. What can we infer about the accuracy of our classifier's performance on those 50 patients? Circle one below and provide your justification.

At most 95% Exactly 95% At least 95% Cannot be determined

(b) We want a more comprehensive model and decides to add more attributes of each patient to our linear classifier. What can we most likely infer about the accuracy of our classifier on the original dataset? Circle one below and provide your justification.

At most 95% Exactly 95% At least 95% Cannot be determined

Now that we are done with choosing the best error (cost) function for our linear classifier and are satisfied with its accuracy on the dataset, we have decided to dig more into the actual process of diagnosing the disease. Surprisingly, it turns out that part of the diagnosis process can be further refined using the least squares method we have learned in class!

To give a bit more context, part of the diagnosis process requires measuring neural impulses from the patient. For the simplicity of this problem, we assume the neural impulses can be measured through a sensor that converts the impulses to **discrete-time signals**. When converting the electric impulses to samples of signals and transmitting these samples to the measurement device, some samples got lost or corrupted in the process. To complicate the problem even more, the missing samples may be randomly distributed through out the signal. Our task is to fill in missing values **based on the available uncorrupted data** in order to conceal these errors (this process is formally known as *error concealment*).

4. To help us reformulate this problem as a least squares problem, we need to derive the matrix properly representing our data. The following subparts will guide you through this process.

(a) Suppose we originally have a total of 5 samples in the signal, and during the transmission process, the 3rd and 4th samples have become unusable (either lost or corrupted). In other words, given a length 5 vector $\vec{s}$ where each entry represents a sample of the signal, $\vec{s}$ is reduced down to a length 3 vector $\vec{r}$ because the original 3rd and 4th entries can no longer be used.

$$\vec{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix} \longrightarrow \begin{bmatrix} s_1 \\ s_2 \\ s_5 \end{bmatrix} = \vec{r}.$$

Find a matrix H that when applied to $\vec{s}$, can result in $\vec{r}$. In other words, find H such that $H\vec{s} = \vec{r}$.

(b) Now, let's think about the reversed process. In the context of this question, we end up receiving the signal only containing the original 1st, 2nd and 5th entries, but we would like to recover our original signal. To "re-expand" the vector back to length 5, for now, we will fill all the missing entries with 0. In other words, we have:

$$\vec{r} = \begin{bmatrix} s_1 \\ s_2 \\ s_5 \end{bmatrix} \longrightarrow \begin{bmatrix} s_1 \\ s_2 \\ 0 \\ 0 \\ s_5 \end{bmatrix} = \vec{s}'.$$

Find a matrix H' such that $H'\vec{r} = \vec{s}'$, and show that $H' = H^T$.

(c) Using the matrix H' we have found in the previous part, we now want to find the "*complement*" of H' (we will call it H'_c). We define the H'_c as the matrix that, when applied, expands a length 2 vector $\vec{r}_c$

containing the two missing entries in $\vec{s}$ to a length 5 vector padded with zeros. In other words, we have:

$$\vec{r}_c = \begin{bmatrix} s_3 \\ s_4 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ 0 \\ s_3 \\ s_4 \\ 0 \end{bmatrix} = \vec{s}_c'.$$

Find the matrix H_c' such that $H_c'\vec{r}_c = \vec{s}_c'$. For the consistency in notation for the rest of this problem, we will also re-express $H_c' = H_c^T$ (to align with H^T from the previous part).

(d) Now that we have found the matrix H^T and its complement H_c^T, we would like to re-express our original signal vector $\vec{s}$. Rewrite the original signal vector $\vec{s}$ in terms of H^T, H_c^T, $\vec{r}$, and $\vec{r}_c$. Within this expression, what is the **unknown term** we are trying to recover?

5. To complete the reformulation of this problem in a least squares setting, we would like to find an expression representing the cost of our model. Suppose we have an orthogonal square matrix D. Given our received signal $\vec{r}$ (has a length of 5; contains missing/corrupted entries as well), our original signal $\vec{s}$, and the matrix D, consider the following expression as the cost function we would like to minimize:

$$C(\vec{s}, \vec{r}) = \|\vec{s} - \vec{r}\|^2 + \|D\vec{s}\|^2$$

Specifically, for the term $\|D\vec{s}\|$, show that $\|D\vec{s}\| = \|\vec{s}\|$ and explain how it helps control the minimization process as when its value increases v.s. decreases.