EECS 16A Designing Information Devices and Systems I
Spring 2020 Homework 13

This homework is due May 1, 2020, at 23:59.
Self-grades are due May 4, 2020, at 23:59.

Submission Format

Your homework submission should consist of **one** file.

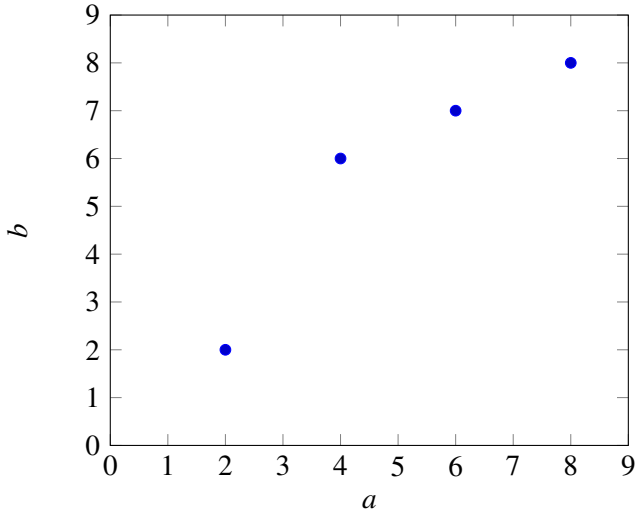
- `hw13.pdf`: A single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

  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. Assign the IPython printout to the correct problem(s) on Gradescope.

Submit the file to the appropriate assignment on Gradescope.

1. Mechanical: Least Squares

The goal of this problem is to use least squares to fit different models (i.e. equations) to a data set. Depending on the model's number of parameters, the model will fit the data better or worse. A better model results in a lower squared error than a worse one. In part (a), we consider a linear model that contains a single slope parameter and intercepts the vertical axis at zero. In part (b), we consider a linear model with a possibly non-zero vertical axis intercept parameter, also known as an affine model.



a	2	4	6	8
b	2	6	7	8

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

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

that best fits the data, where x is a scalar that minimizes the squared error

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

Note: By using this linear model, we are implicitly forcing the line to go through the origin.

You may use a calculator but show your work. Do not directly plug your numbers into IPython. Once you've computed the optimal solution $\hat{\vec{x}}$, compute the squared error between your model's prediction and the actual b values as shown in Equation 1. Plot the best fit line along with the data points to examine the quality of the fit. You may either use the provided IPython notebook code to plot your best fit line or hand draw it.

Solution:

Define $\vec{a} = \begin{bmatrix} 2 & 4 & 6 & 8 \end{bmatrix}^T$ and $\vec{b} = \begin{bmatrix} 2 & 6 & 7 & 8 \end{bmatrix}^T$. Applying the linear least squares formula, we get

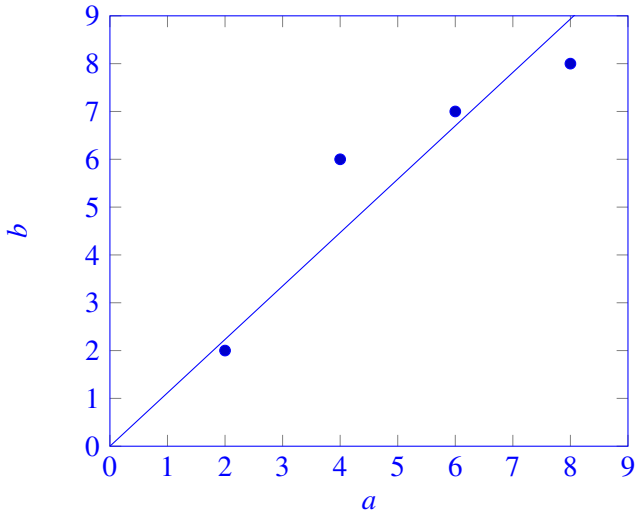
$$\begin{aligned} \hat{\vec{x}} &= (\vec{a}^T\vec{a})^{-1}\vec{a}^T\vec{b} \\ &= \left(\begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}^T \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix} \right)^{-1} \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}^T \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} \\ &= (120)^{-1}(134) = 1.1167 \end{aligned}$$

The error between the model's prediction $\hat{b}$ and actual values b is

$$\begin{aligned} \vec{e} &= \hat{\vec{b}} - \vec{b} = \vec{a}\hat{\vec{x}} - \vec{b} \\ &= 1.1167 \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix} - \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 0.234 \\ -1.534 \\ -0.3 \\ 0.934 \end{bmatrix} \end{aligned}$$

and the sum of squared errors is

$$\vec{e}^T\vec{e} = 3.367$$



(b) Let us consider a model with a (vertical) b-intercept. That is, we can get a better fit for the data by assuming a(n) (affine) model of the form

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

Set up a least squares problem to find the optimal x_1 and x_2 and compute the squared error between your model's prediction and the actual $\vec{b}$ values. Plot your affine model. Is it a better fit for the data? Support your answer both qualitatively by examining how close the best fit lines are to the data points and quantitatively by providing a numerical justification.

Solution:

Let $\vec{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. Using the least squares formula with the new augmented $\mathbf{A}$ matrix, we calculate the optimal approximation of $\vec{x}$ as

$$\begin{aligned}
\hat{\vec{x}} &= \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{b} \\
&= \left(\begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix}\right)^{-1} \begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix}^T \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} \\
&= \begin{bmatrix} 120 & 20 \\ 20 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 2 & 4 & 6 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ 7 \\ 8 \end{bmatrix} \\
&= \frac{1}{120(4) - 20(20)} \begin{bmatrix} 4 & -20 \\ -20 & 120 \end{bmatrix} \begin{bmatrix} 134 \\ 23 \end{bmatrix} \\
\hat{\vec{x}} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} &= \begin{bmatrix} 0.95 \\ 1 \end{bmatrix}
\end{aligned}$$

The linear model's prediction of $\vec{b}$ is given by

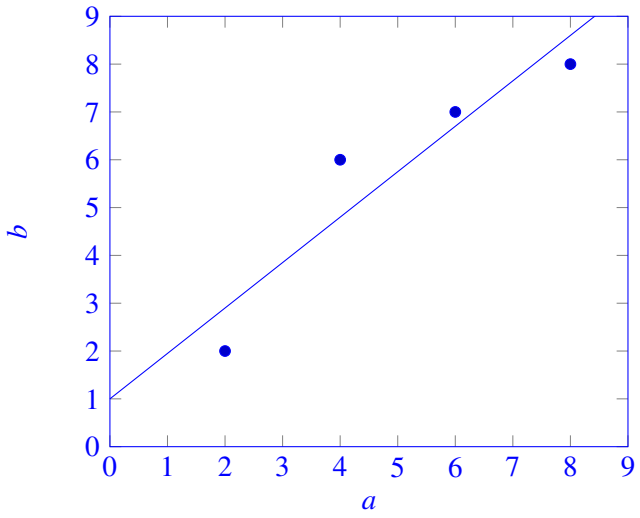
$$\hat{\vec{b}} = \mathbf{A}\hat{\vec{x}} = \begin{bmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \\ 8 & 1 \end{bmatrix} \begin{bmatrix} 0.95 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.9 \\ 4.8 \\ 6.7 \\ 8.6 \end{bmatrix}$$

and the error is given by

$$\vec{e} = \hat{\vec{b}} - \vec{b} = \begin{bmatrix} 0.9 & -1.2 & -0.3 & 0.6 \end{bmatrix}^T$$

The summed squared error is

$$\vec{e}^T\vec{e} = 2.7$$



We can see qualitatively from the plot that the line passes closer to the data points than the best fit line found previously. We see quantitatively that the the sum of the squared errors is lower than that of the model found in part (a).

(c) **Prove the following theorem.**

Let $\mathbf{A} \in \mathbb{R}^{m,n}$. If $\hat{\vec{x}}$ is the solution to the least squares problem

$$\min_{\vec{x}} \left\| \mathbf{A}\vec{x} - \vec{b} \right\|^2,$$

then $\mathbf{A}^T(\mathbf{A}\hat{\vec{x}} - \vec{b}) = \vec{0}$. This is called the normal equation; it says that the error in the least squares estimate is orthogonal to the columns of $\mathbf{A}$. You will often see the normal equation written in the form $\mathbf{A}^T\mathbf{A}\vec{x} = \mathbf{A}^T\vec{b}$.

Solution:

Plugging in the linear least squares formula for $\hat{\vec{x}}$, we get

$$\begin{aligned} \langle \mathbf{A}, \mathbf{A}\hat{\vec{x}} - \vec{b} \rangle &= \mathbf{A}^T\left(\mathbf{A}\hat{\vec{x}} - \vec{b}\right) \\ &= \mathbf{A}^T\left(\mathbf{A}\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{b} - \vec{b}\right) \\ &= \mathbf{A}^T\mathbf{A}\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{b} - \mathbf{A}^T\vec{b} \\ &= \mathbf{I}\mathbf{A}^T\vec{b} - \mathbf{A}^T\vec{b} \\ &= \mathbf{A}^T\vec{b} - \mathbf{A}^T\vec{b} = \vec{0} \end{aligned}$$

2. Trilateration With Noise!

In this question, we will explore how various types of noise affect the quality of triangulating a point on the 2D plane to see when trilateration works well and when it does not.

First, we will remind ourselves of the fundamental equations underlying trilateration.

(a) There are four beacons at the known coordinates $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$. You are located at some unknown coordinate (x, y) that you want to determine. The distance between your location and each of the four beacons are d_1 through d_4, respectively. Write down one equation for each beacon that relates the coordinates to the distances using the Pythagorean Theorem.

Solution: For each beacon, we have the equation

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2,$$

for $i \in \{1, 2, 3, 4\}$.

(b) Unfortunately, the above system of equations is nonlinear, so we can't use least squares or Gaussian Elimination to solve it. We will use the technique discussed in lecture to obtain a system of linear equations. In particular, we can subtract the first of the above equations from the other three to obtain three linear equations. Write down these three linear equations.

Solution: Subtracting the 1st equation from the ith, we obtain

$$(x - x_i)^2 - (x - x_1)^2 + (y - y_i)^2 - (y - y_1)^2 = d_i^2 - d_1^2,$$

so expanding and canceling the x^2 and y^2 terms, we obtain

$$-2xx_i + x_i^2 + 2xx_1 - x_1^2 - 2yy_i + y_i^2 + 2yy_1 - y_1^2 = d_i^2 - d_1^2$$

for $i \in \{2, 3, 4\}$.

(c) Combine the three equations in the above system into a single matrix equation of the form

$$\mathbf{A} \begin{bmatrix} x \\ y \end{bmatrix} = \vec{b}.$$

Solution: Rearranging each of the above equations, we obtain

$$(-2x_i + 2x_1)x + (-2y_i + 2y_1)y = (d_i^2 - x_i^2 - y_i^2) - (d_1^2 - x_1^2 - y_1^2)$$

for $i \in \{2, 3, 4\}$. Stacking and writing in matrix form, we obtain

$$\begin{bmatrix} 2(-x_2 + x_1) & 2(-y_2 + y_1) \\ 2(-x_3 + x_1) & 2(-y_3 + y_1) \\ 2(-x_4 + x_1) & 2(-y_4 + y_1) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (d_2^2 - x_2^2 - y_2^2) - (d_1^2 - x_1^2 - y_1^2) \\ (d_3^2 - x_3^2 - y_3^2) - (d_1^2 - x_1^2 - y_1^2) \\ (d_4^2 - x_4^2 - y_4^2) - (d_1^2 - x_1^2 - y_1^2) \end{bmatrix}.$$

(d) Now, go to the IPython notebook. In the notebook we are given three possible sets of measurements for the distances of each beacon from the receiver:

i. `ideal_distances`: the ideal set of measurements, the true distances of our receiver to the beacons. $d_1 = d_2 = d_3 = d_4 = 5$.

ii. `imperfect_distances`: imperfect measurements. $d_1 = 5.5, d_2 = 4.5, d_3 = 5, d_4 = 5$.

iii. `one_bad_distances`: mostly perfect measurements, but d_1 is a very bad measurement. $d_1 = 6.5$ and $d_2 = d_3 = d_4 = 5$.

Plot the graph illustrating the case when the receiver has received `ideal_distances` and visually solve for the position of the observer (x, y). What is the coordinate?

Solution: From the plot, it is clear that $(x, y) = (0, 0)$, since all four circles intersect at that point.

(e) You will now set up the above linear system using IPython. Fill in each element of the matrix $\mathbf{A}$ that you found in part (c).

Solution: $\mathbf{A} = \begin{bmatrix} 2(-x_2 + x_1) & 2(-y_2 + y_1) \\ 2(-x_3 + x_1) & 2(-y_3 + y_1) \\ 2(-x_4 + x_1) & 2(-y_4 + y_1) \end{bmatrix}$.

See the IPython notebook for the actual code.

(f) Similarly, fill in the entries of $\vec{b}$ from part (c) in the `make_b` function.

Solution: $\vec{b} = \begin{bmatrix} (d_2^2 - x_2^2 - y_2^2) - (d_1^2 - x_1^2 - y_1^2) \\ (d_3^2 - x_3^2 - y_3^2) - (d_1^2 - x_1^2 - y_1^2) \\ (d_4^2 - x_4^2 - y_4^2) - (d_1^2 - x_1^2 - y_1^2) \end{bmatrix}$.

See the IPython notebook for the actual code.

(g) Now, you should be able to plot the estimated position of (x, y) using the supplied code for the `ideal_distances` observations. Modify the code to estimate (x, y) for `imperfect_distances` and `one_bad_distances`, and comment on the results.

In particular, for `one_bad_distances` would you intuitively have chosen the same point that our trilateration solution did knowing that only one measurement was bad?

Solution: We see that the solution to (x, y) moves away from the origin in the latter two cases. For `one_bad_distances`, even through three out of the four circles intersect at the origin (suggesting that $(x, y) = (0, 0)$), our least squares approach picks a point away from the origin, indicating that it might not be determining the best solution possible.

(h) We define the "cost" of a position (x, y) to be the sum of the squares of the differences in distance of that position from the observation, as defined symbolically in the notebook. Study the heatmap of the cost of various positions on the plane, and make sure you see why $(0, 0)$ appears to be the point with the lowest cost.

Now, compare the cost of $(0, 0)$ with the cost of your estimated position obtained from the least-squares solution in all three cases. When does least squares do worse?

Solution: See IPython solutions. For `ideal_distances`, both approaches yield a cost of 0.0. For `imperfect_distances`, $(0, 0)$ is actually slightly worse than our least-squares solution, but in both cases the costs are fairly low.

For `one_bad_distances`, the costs are lower at $(0, 0)$ compared to the least-squares solution, as we expected from the heatmap.

3. Labeling Patients Using Gene Expression Data

Least squares techniques are useful for many different kinds of prediction problems. Numerous researchers have extensively further developed the core ideas that we have learned in class. These ideas are commonly used in machine learning for finance, healthcare, advertising, image processing, and many other fields. Here, we'll explore how least squares can be used for classification of data in a medical context.

Gene expression data of patients, along with other factors such as height, weight, age, and family history, are often used to predict the likelihood that a patient might develop a certain disease. This data can be combined into a vector that describes each patient. This vector is often referred to as a feature vector.

Many scientific studies examine mice to understand how gene expression relates to diabetes in humans. Studies have shown that the expression of the tomosin2 and ts1 genes are correlated to the onset of diabetes in

mice. How can we predict whether or not a mouse will develop diabetes based on data about this expression as well as other factors of the mouse? We will use some (fake) data to explore this.

We are given feature vectors for each mouse as:

$$\begin{bmatrix} \text{age} \\ \text{weight} \\ \text{tomosin2} \\ \text{ts1} \\ \text{chn1} \end{bmatrix}$$

Age and weight in the vector above are represented by real numbers, while the presence or absence of the expression of the genes tomosin2, ts1, and chn1 is captured by $+1$ and -1 respectively. For example, the vector $\begin{bmatrix} 2 & 20 & 1 & -1 & -1 \end{bmatrix}^T$ means a 2 month old mouse that weighs 20 grams and expresses the genes tomosin2 but not ts1 or chn1.

We would like the following expression to be positive if the mouse has diabetes and negative if the mouse does not have diabetes:

$$f(\text{age}, \text{weight}, \text{tomoson2}, \text{ts1}, \text{chn1}) = \alpha_1(\text{age}) + \alpha_2(\text{weight}) + \alpha_3(\text{tomosin2}) + \alpha_4(\text{ts1}) + \alpha_5(\text{chn1}).$$

(a) We wish to set up a linear model for the problem in the format $\mathbf{A}\vec{x} = \vec{b}$. Here, $\vec{b}$ will be a vector with $+1, -1$ entries where a 1 represents that the mouse is diabetic and -1 represents that the mouse is not diabetic. The feature vectors of each mouse will be included in the rows of the matrix $\mathbf{A}$. Set up the problem by writing $\mathbf{A}$, $\vec{x}$, and $\vec{b}$ in terms of the variables in the feature vectors, α_i, and any other variables you define. What are your unknowns?

Solution: Assume we have n mice. Define b_j as the indicator for whether the j-th mouse has diabetes. Each of the data variables will be indexed by $j = 1, \ldots, n$ for each of the n mice. The unknowns that we want to find are α_i, where $\{i = 1, \ldots, 5\}$. The vector $\vec{x}$ is the vector of unknowns $\vec{x} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \end{bmatrix}^T$. The complete problem setup is:

$$\underbrace{\begin{bmatrix} \text{age}_1 & \text{weight}_1 & \text{tomosin2}_1 & \text{ts1}_1 & \text{chn1}_1 \\ \text{age}_2 & \text{weight}_2 & \text{tomosin2}_2 & \text{ts1}_2 & \text{chn1}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{age}_n & \text{weight}_n & \text{tomosin2}_n & \text{ts1}_n & \text{chn1}_n \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{\vec{b}}$$

(b) Training data is data that is used to develop your model. Use the (fake) *training* data `diabetes_train.npy` to find the optimal model parameters for the given data set. What are the optimal parameter values? Use the provided IPython notebook.

Solution:

To solve for $\vec{x}$ in $\mathbf{A}\vec{x} = \vec{b}$ using the least squares technique, we find $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{b}$. The result is

$$\vec{x} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} 0.12131475 \\ -0.15253102 \\ -0.36111722 \\ -0.06427341 \\ 0.95936096 \end{bmatrix}$$

(c) Now it is time to use the model you have developed to make some predictions! It is interesting to note here that we are not looking for a real number to model whether each mouse has diabetes or not; we are looking for a binary label. Therefore, we will use the *sign* of the expression above to assign a ± 1 value to each mouse. Predict whether each mouse with the characteristics in the *test* data set `diabetes_test.npy` will get diabetes. There are four mice in the test data set. Include the ± 1 vector that indicates whether or not they have diabetes in your answer. What is the prediction accuracy (number of correct predictions divided by total number of predictions) of your model?

Solution:

Using the values of α_i calculated in the previous part on the test data, we see that the prediction is $\vec{b} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$, which differs in the third entry of the given file `diabetes_test.npy`. Therefore the prediction accuracy is 75%.

4. Image Analysis

Applications in medical imaging often require an analysis of images based on the image's pixels. For instance, we might want to count the number of cells in a given sample. One way to do this is to take a picture of the cells and use the pixels to determine their locations and how many there are. Automatic detection of shape is useful in image classification as well (e.g. consider a robot trying to find out autonomously where a mug is in its field of vision).

Let us focus back on the medical imaging scenario. You are interested in finding the exact position and shape of a cell in an image. You will do this by finding the equation of the circle or ellipse that bounds the cell relative to a given coordinate system in the image. Your collaborator uses edge detection techniques to find a bunch of points that are approximately along the edge of the cell. We assume that the origin is in the center of the image with standard axes (x, y) and collect the following points:

$(0.3, -0.69)$, $(0.5, 0.87)$, $(0.9, -0.86)$, $(1, 0.88)$, $(1.2, -0.82)$, $(1.5, 0.64)$, $(1.8, 0)$.

Recall that an equation of the form

$$ax^2 + bxy + cy^2 + dx + ey = 1$$

can be used to represent an ellipse (if $b^2 - 4ac < 0$), and an equation of the form

$$a(x^2 + y^2) + dx + ey = 1$$

is a circle if $d^2 + e^2 + 4a > 0$. The circle has fewer parameters.

(a) How can you find the equation of a circle that surrounds the cell? First, provide a setup and formulate a minimization problem to do this, i.e. a least squares problem minimizing the squared error $||\mathbf{A}\vec{x} - \vec{b}||$ where you attempt to find the unknown coefficients a, d, and e from your points. *Hint: The quantities $(x^2 + y^2)$, x, and y can be thought of as variables calculated from your data points.*

Solution:

The setup is:

$$\min_{a,d,e} \left\| \begin{bmatrix} x^2 + y^2 & x & y \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ d \\ e \end{bmatrix} - \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right\|$$

We plug in numbers to get:

$$\min_{a,d,e} \left\| \begin{bmatrix} 0.5661 & 0.3 & -0.69 \\ 1.0069 & 0.5 & 0.87 \\ 1.5496 & 0.9 & -0.86 \\ 1.7744 & 1 & 0.88 \\ 2.1124 & 1.2 & -0.82 \\ 2.6596 & 1.5 & 0.64 \\ 3.24 & 1.8 & 0 \end{bmatrix} \begin{bmatrix} a \\ d \\ e \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\|$$

(b) How can you find the equation of an ellipse that surrounds the cell? Provide a setup and formulate a minimization problem similar to that in part (a).

Solution:

The setup is:

$$\min_{a,b,c,d,e} \left\| \begin{bmatrix} x^2 & xy & y^2 & x & y \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} - \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right\|$$

We then plug in values to get:

$$\min_{a,b,c,d,e} \left\| \begin{bmatrix} 0.09 & -0.207 & 0.4761 & 0.3 & -0.69 \\ 0.25 & 0.435 & 0.7569 & 0.5 & 0.87 \\ 0.81 & -0.774 & 0.7396 & 0.9 & -0.86 \\ 1 & 0.88 & 0.7744 & 1 & 0.88 \\ 1.44 & -0.984 & 0.6724 & 1.2 & -0.82 \\ 2.25 & 0.96 & 0.4096 & 1.5 & 0.64 \\ 3.24 & 0 & 0 & 1.8 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\|$$

(c) In the IPython notebook, write a short program to fit a circle to the given points. What is $\frac{\|\vec{e}\|}{N}$, where $\vec{e} = \mathbf{A}\vec{x} - \vec{b}$ and N is the number of data points? Plot your points and the best fit circle in IPython.

Solution:

See the IPython notebook.

The solution vector is:

$$\vec{x} = \begin{bmatrix} 4.87 \\ -7.89 \\ -0.23 \end{bmatrix}$$

Thus, we would predict the equation of the circle to be: $4.87(x^2 + y^2) - 7.89x - 0.23y = 1$.
This gives the normalized error: $\frac{0.96}{7} = 0.137$.

(d) In the IPython notebook, write a short program to fit an ellipse to the given points. What is $\frac{\|\vec{e}\|}{N}$, where $\vec{e} = \mathbf{A}\vec{x} - \vec{b}$ and N is the number of data points? Plot your points and the best fit ellipse in IPython. How does this error compare to the one in the previous subpart? Which technique is better?

Solution:

See the IPython notebook.

The solution vector is:

$$\vec{x} = \begin{bmatrix} 4.10 \\ 0.49 \\ 4.94 \\ -6.85 \\ -0.62 \end{bmatrix}$$

We predict the general equation to be: $4.10x^2 + 0.49xy + 4.94y^2 - 6.85x - 0.62y = 1$.
This gives the normalized error: $\frac{0.090}{7} = 0.0128$.

The ellipse is a better fit because it has more parameters, so the least squares technique can tune the parameters to be closer to the observations.

5. Constrained Least Squares Optimization

In this problem, we will guide you through solving the following optimization problem:

Consider a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ where $M > N$ and all N columns are linearly independent. Determine a unit vector $\hat{\vec{x}}$ that minimizes $\|\mathbf{A}\vec{x}\|$, where $\|\cdot\|$ denotes the norm—that is,

$$\|\mathbf{A}\vec{x}\|^2 \triangleq \langle \mathbf{A}\vec{x}, \mathbf{A}\vec{x} \rangle = (\mathbf{A}\vec{x})^T \mathbf{A}\vec{x} = \vec{x}^T \mathbf{A}^T \mathbf{A}\vec{x}.$$

This is equivalent to solving the following optimization problem:

$$\min_{\vec{x}} \|\mathbf{A}\vec{x}\|^2 \quad \text{subject to the constraint} \quad \|\vec{x}\|^2 = 1.$$

This task may *seem* like solving a standard least squares problem $\mathbf{A}\vec{x} = \vec{b}$ where $\vec{b} = \vec{0}$, but it is different. As an example, notice $\vec{x} = \vec{0}$ is *not* a valid solution to our problem because the norm of the zero vector does not equal one. Our optimization problem is a least squares problem with a constraint—hence the term *constrained least squares optimization*. The constraint can be visualized as limiting the vector $\vec{x}$ to lie on a unit circle (radius of the circle is one) if $N = 2$ and on a unit sphere if $N = 3$.

Let $(\lambda_1, \vec{v}_1), \ldots, (\lambda_N, \vec{v}_N)$ denote the eigenpairs (i.e., eigenvalue/eigenvector pairs) of $\mathbf{A}^T\mathbf{A}$. Assume that the eigenvalues are all real, distinct and indexed in an descending fashion—that is,

$$\lambda_1 > \cdots > \lambda_N.$$

Assume, too, that each eigenvector has been normalized to have unit length—that is, $\|\vec{v}_k\| = 1$ for all $k \in \{1, \ldots, N\}$.

(a) Show that $\lambda_N > 0$, i.e. all the eigenvalues are strictly positive.
Hint: Consider $\|\mathbf{A}\vec{v}\|^2$
Solution:
Consider $\|\mathbf{A}\vec{v}\|^2$ for eigenvector $\vec{v}$, with eigenvalue λ.

$$\begin{aligned} \|\mathbf{A}\vec{v}\|^2 &= \vec{v}^T (\mathbf{A}^T \mathbf{A}\vec{v}) \\ &= \vec{v}^T \lambda \vec{v} \\ \lambda &= \frac{\|\mathbf{A}\vec{v}\|^2}{\|\vec{v}\|^2} \end{aligned}$$

Therefore, $\lambda > 0$ since norms are positive if $\vec{v} \neq \vec{0}$. Since the columns of $\mathbf{A}$ are all linearly independent, the numerator is never zero unless $\vec{v} = \vec{0}$. Recall that even though the zero vector always satisfies the definition of an eigenvector, we never consider it to be an eigenvector because it is a trivial solution.

(b) Consider two eigenpairs $(\lambda_k, \vec{v}_k)$ and $(\lambda_\ell, \vec{v}_\ell)$ corresponding to distinct eigenvalues of $\mathbf{A}^T\mathbf{A}$—that is, $\lambda_k \neq \lambda_\ell$. Prove that the corresponding eigenvectors $\vec{v}_k$ and $\vec{v}_\ell$ are orthogonal: $\vec{v}_k \perp \vec{v}_\ell$.

To help you get started, consider the two equations

$$\mathbf{A}^T\mathbf{A}\,\vec{v}_k = \lambda_k\,\vec{v}_k \tag{2}$$

and

$$\vec{v}_\ell^{\,T}\mathbf{A}^T\mathbf{A} = \lambda_\ell\,\vec{v}_\ell^{\,T}. \tag{3}$$

The second equation can be derived by taking the transpose of both sides in the eigenvalue equation $\mathbf{A}^T\mathbf{A}\,\vec{v}_\ell = \lambda_\ell\,\vec{v}_\ell$. Premultiply Equation 2 with $\vec{v}_\ell^{\,T}$, postmultiply Equation 3 with $\vec{v}_k$, compare the two, and explain how one may then infer that $\vec{v}_k$ and $\vec{v}_\ell$ are orthogonal, i.e. $\langle \vec{v}_k, \vec{v}_\ell \rangle = 0$.

Premultiplication by a vector $\vec{x}$ means multiplying an expression by $\vec{x}$ on the left. For example, pre-multiplying the matrix $\mathbf{A}$ by $\vec{x}$ gives $\vec{x}\mathbf{A}$. Postmultiplication means multiplying on the right. Remember that in general matrix-vector multiplication is not commutative, so those operations are not identical.

Solution:

Following the hint:

$$\vec{v}_\ell^{\,T}\mathbf{A}^T\mathbf{A}\,\vec{v}_k = \vec{v}_\ell^{\,T}\lambda_k\,\vec{v}_k$$
$$\vec{v}_\ell^{\,T}\mathbf{A}^T\mathbf{A}\vec{v}_k = \lambda_\ell\,\vec{v}_\ell^{\,T}\,\vec{v}_k$$

We see that the two expressions on the left are equal, so we set the two expressions on the right equal to each other:

$$\lambda_k\vec{v}_\ell^{\,T}\,\vec{v}_k = \lambda_\ell\,\vec{v}_\ell^{\,T}\,\vec{v}_k$$

If $\lambda_k \neq \lambda_l$, then the only possible solution is that $\vec{v}_\ell^{\,T}\,\vec{v}_k = 0$, which means $\vec{v}_\ell$ and $\vec{v}_k$ are orthogonal.

(c) The results of part (b) imply that the N eigenvectors of $\mathbf{A}^T\mathbf{A}$ are mutually orthogonal. A basis formed by vectors that are both (1) mutually orthogonal and (2) have unit length is called an orthonormal basis. Since the eigenvalues of $\mathbf{A}^T\mathbf{A}$ are distinct and have a norm of one, the eigenvectors form an orthonormal basis in $\mathbb{R}^N$. This means that we can express an arbitrary vector $\vec{x} \in \mathbb{R}^N$ as a linear combination of the eigenvectors $\vec{v}_1, \ldots, \vec{v}_N$, as follows:

$$\vec{x} = \sum_{n=1}^{N} \alpha_n\,\vec{v}_n.$$

i. Determine the n^{th} coefficient α_n in terms of $\vec{x}$ and one or more of the eigenvectors $\vec{v}_1, \ldots, \vec{v}_N$.

Solution:

Since $\vec{v}_n$ are orthogonal, the coefficient α_n is the projection of $\vec{x}$ on to $\vec{v}_n$.

Since $\vec{v}_n$ are all unit vectors, the projection is simply the inner product.

$$\alpha_n = \langle \vec{x}, \vec{v}_n \rangle = \vec{x}^T\vec{v}_n$$

ii. Suppose $\vec{x}$ is a unit-length vector (i.e., a unit vector) in $\mathbb{R}^N$. Show that

$$\sum_{n=1}^{N} \alpha_n^2 = 1$$

where the α_n's are the coefficients of $\vec{x}$ in the basis defined earlier.

Solution:

Consider $\|\vec{x}\|^2 = 1$.

$$\begin{aligned}\|\vec{x}\|^2 &= \vec{x}^T\vec{x} \\ &= \left(\sum_{i=1}^{N} \alpha_i \vec{v}_i\right)^T \left(\sum_{j=1}^{N} \alpha_j \vec{v}_j\right) = \left(\sum_{i=1}^{N} \alpha_i \vec{v}_i^T\right)\left(\sum_{j=1}^{N} \alpha_j \vec{v}_j\right) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \vec{v}_i^T \vec{v}_j\end{aligned}$$

Now, since the $\vec{v}_i$ are orthogonal, we know: $\vec{v}_i^T\vec{v}_j = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$

Therefore, $\|\vec{x}\|^2 = \sum_{n=1}^{N} \alpha_n^2 = 1$.

(d) Now express $\|\mathbf{A}\vec{x}\|^2$ in terms of $\{\alpha_1, \alpha_2 \ldots \alpha_N\}$, $\{\lambda_1, \lambda_2 \ldots \lambda_N\}$, and $\{\vec{v}_1, \vec{v}_2 \ldots \vec{v}_N\}$, and find an expression for $\vec{x}$ such that $\|\mathbf{A}\vec{x}\|^2$ is minimized. Do *not* use any tool from calculus to solve this problem, so avoid differentiation.

Hint: After expressing $\|\mathbf{A}\vec{x}\|^2$ in terms of $\{\alpha_1, \alpha_2 \ldots \alpha_N\}$, $\{\lambda_1, \lambda_2 \ldots \lambda_N\}$, and $\{\vec{v}_1, \vec{v}_2 \ldots \vec{v}_N\}$, which variable are you minimizing over?

Solution:

Note that $\|\mathbf{A}\vec{x}\|^2 = \vec{x}^T\mathbf{A}^T\mathbf{A}\vec{x}$. We express $\vec{x}$ in terms of $\vec{v}_i$ (the eigenvalues of $\mathbf{A}^T\mathbf{A}$) and expand.

$$\begin{aligned}\mathbf{A}^T\mathbf{A}\vec{x} &= \mathbf{A}^T\mathbf{A}\sum_{n=1}^{N} \alpha_n \vec{v}_n \\ &= \sum_{n=1}^{N} \alpha_n \mathbf{A}^T\mathbf{A}\vec{v}_n \\ &= \sum_{n=1}^{N} \alpha_n \lambda_n \vec{v}_n\end{aligned}$$

Now:

$$\begin{aligned}\vec{x}^T\mathbf{A}^T\mathbf{A}\vec{x} &= \vec{x}^T \sum_{n=1}^{N} \alpha_n \lambda_n \vec{v}_n \\ &= \left(\sum_{n=1}^{N} \alpha_n \vec{v}_n^T\right)\left(\sum_{n=1}^{N} \alpha_n \lambda_n \vec{v}_n\right) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \lambda_j \vec{v}_i^T \vec{v}_j \\ &= \sum_{n=1}^{N} \alpha_n^2 \lambda_n \qquad (4)\end{aligned}$$

We know from part (c) that α_n are constrained by the following equation:

$$\sum_{n=1}^{N} \alpha_n^2 = 1 \tag{5}$$

Equation 4 is a weighted sum of all the eigenvalues. All of the eigenvalues are larger than zero. To minimize that sum under the constraint in equation 5, we want to put as much weight on the smallest eigenvalue λ_N as possible. Therefore, we pick $\alpha_N = 1$ and $\alpha_k = 0$, where $k \in \{1, 3, \ldots, N-1\}$. This is equivalent to picking $\vec{x}$ to be the eigenvector $\vec{v}_N$ with the smallest eigenvalue. The minimum value achieved is λ_N.

6. (PRACTICE) Recipe Reconnaissance

Engineering Edibles has been growing in size and popularity and has introduced two new cookies: Decadent Dwight and Heavenly Hearst. As a result of their popularity there is an increased interest in understanding their secret recipes.

The bakery produces 40 Decadent Dwight and 50 Heavenly Hearst cookies each day. Each cookie costs \$1. The bakery business is way overpriced, and everyone knows about 80 cents of the cost comes from profits (and labor); ingredients are only worth *about* 20 cents.

The team from Berkeley wants to figure out the recipes for the two new cookies, and they know the recipes are *different*. For the purpose of this problem, each cookie only contains three ingredients: eggs, sugar and butter.

There are 6 unknowns: the amount of eggs, sugar, and butter used in Decadent Dwight cookies and the amount of eggs, sugar, and butter used in Heavenly Hearst cookies. We will denote these D_e $\frac{\text{eggs}}{\text{cookie}}$, D_s $\frac{\text{grams sugar}}{\text{cookie}}$, D_b $\frac{\text{grams butter}}{\text{cookie}}$, H_e $\frac{\text{eggs}}{\text{cookie}}$, H_s $\frac{\text{grams sugar}}{\text{cookie}}$, H_b $\frac{\text{grams butter}}{\text{cookie}}$.

(a) How many linearly independent equations would you need to be able to solve for all the unknowns (assuming a consistent system of equations)?

Solution: You would need 6 linearly independent equations, if the system is consistent.

(b) Unfortunately, the team is not able to find precise information about the ingredients used in the cookie making process.

They stake-out Engineering Edibles, and they see Bob the Baker buy a dozen eggs for \$2 **daily** and a 5 kg bag of sugar **every week**. They hire a master-taster, who tells them there is **exactly** 10 grams of butter in each of the cookies (for both Decadent Dwight and Heavenly Hearst). They know from the supermarket that **exactly** 1 kg of sugar costs 5 dollars, and **exactly** 100 grams of butter costs 1 dollar. (*The team precisely knows amount of butter in each cookie and the exact price of sugar and butter. However, the amount of eggs and sugar in each cookie are imprecisely known.*)

All this is not enough for the team to unlock the secret recipe! One day, however, the team gets lucky and hears through the grapevine that Heavenly Hearst contains **about** 10 grams of sugar. (*This value is not precise, since it's just gossip!*)Let's also remember that the ingredients for each cookie of either variety would cost \$0.2.

Using the information above, set up the problem as a least-squares problem and find best estimate of the secret recipe. Identify what variables you are estimating. You are welcome to use a computer to solve this.

Solution:

Rubric Note: Give yourself full credit as long as you set up the least squares problem correctly, you don't have to worry about scaling changing the numerical solutions.

We know from the master-taster that $D_b = 10 \frac{\text{grams}}{\text{cookie}}$ and $H_b = 10 \frac{\text{grams}}{\text{cookie}}$. This leaves us with only 4 unknowns.

We set up the following system of equations:

$$40 \times D_e + 50 \times H_e = 12 \tag{6}$$
$$7 \times (40 \times D_s + 50 \times H_s) = 5000 \tag{7}$$
$$\frac{2}{12} \times D_e + \frac{5}{1000} \times D_s = 0.10 \tag{8}$$
$$\frac{2}{12} \times H_e + \frac{5}{1000} \times H_s = 0.10 \tag{9}$$
$$H_s = 10 \tag{10}$$

i. **Eggs**: Bob buys a dozen eggs every day, and he makes 40 Decadent Dwight cookies and 50 Heavenly Hearst cookies daily. Therefore the 12 eggs must be somehow divided into the cookies! 40 cookies $\times D_e \frac{\text{eggs}}{\text{cookie}} + 50$ cookies $\times H_e \frac{\text{eggs}}{\text{cookie}} = 12$ eggs

ii. **Sugar**: Similar logic as Equation 1, except he buys 5000 grams of sugar each week. 7 days $\times$ $(40 \frac{\text{cookies}}{\text{day}} \times D_s \frac{\text{grams}}{\text{cookie}} + 50 \frac{\text{cookies}}{\text{day}} \times H_s \frac{\text{grams}}{\text{cookie}}) = 5000$ grams

iii. **Cost**: We're told that each cookie uses about 20 cents worth of ingredients. We first subtract the cost of butter, which is 10 grams $\times \frac{\$1}{100 \text{ grams}} = \$0.1$. Now we know the eggs and sugar in the cookie cost 20 - 10 = 10 cents. The rest follows from dimensional analysis: $\frac{\$2}{12 \text{ eggs}} \times D_e \frac{\text{eggs}}{\text{cookie}} + \frac{\$5}{1000 \text{ grams}} \times D_s \frac{\text{grams}}{\text{cookie}} = 0.1 \frac{\$}{\text{cookie}}$

iv. **Cost**: Same as Equation 3, except for Heavenly Hearst. This symmetry results from the ingredients costing the same regardless of cookie, and the cookies being sold for the same price!

v. **Sugar Estimate,** H_s: We are given this information!

Now we can simplify and set this up in a matrix:

$$\begin{bmatrix} 40 & 50 & 0 & 0 \\ 0 & 0 & 40 & 50 \\ \frac{2}{12} & 0 & \frac{5}{1000} & 0 \\ 0 & \frac{2}{12} & 0 & \frac{5}{1000} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} D_e \\ H_e \\ D_s \\ H_s \end{bmatrix} = \begin{bmatrix} 12 \\ \frac{5000}{7} \\ 0.1 \\ 0.1 \\ 10 \end{bmatrix}$$

Now we can use iPython to solve using least squares (see sol13.ipynb). This will return the following solution vector:

$$\begin{bmatrix} D_e \\ H_e \\ D_s \\ H_s \end{bmatrix} = \begin{bmatrix} 0.2386 \\ 0.0491 \\ 5.3571 \\ 10 \end{bmatrix}$$

(c) The Berkeley team decides to do scientific experiments to better their estimates of the recipe. They obtain a scale, and weigh the new cookies. Their (cheap and second-hand) scale is only accurate to the gram and easily affected by air currents, so they get noisy observations: Decadent Dwight is **about** 25 grams, and Heavenly Hearst is **about** 24 grams. Assume that 1 egg weighs **exactly** 50 grams.

Update the least squares problem with this new information, and see how the values change. You should formulate two new equation in addition to the equations from the last part.

Are the new values more accurate? (*Hint: Find the sum of squared errors,* $||e||^2$ *then divide* $||e||$ *by the number of measurements to find the* ***average error for data points*** *for both (b) and (c).*)

Solution:

Rubric Note: Give yourself full credit as long as you set up the least squares problem correctly, you don't have to worry about scaling changing the numerical solutions.

To use the new information, we simply update the previous matrix:

$$\begin{bmatrix} 40 & 50 & 0 & 0 \\ 0 & 0 & 40 & 50 \\ \frac{2}{12} & 0 & \frac{5}{1000} & 0 \\ 0 & \frac{2}{12} & 0 & \frac{5}{1000} \\ 0 & 0 & 0 & 1 \\ 50 & 0 & 1 & 0 \\ 0 & 50 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} D_e \\ H_e \\ D_s \\ H_s \end{bmatrix} = \begin{bmatrix} 12 \\ \frac{5000}{7} \\ 0.1 \\ 0.1 \\ 10 \\ 15 \\ 14 \end{bmatrix}$$

Again, we first subtract the mass of butter (in this case, 10 grams) from the measurements. Since we know 1 egg = 50 grams, and since sugar is already given in units of grams, the equation follows:
$50 \ \frac{\text{grams}}{\text{egg}} \times D_e \ \frac{\text{eggs}}{\text{cookie}} + D_s \ \frac{\text{grams}}{\text{cookie}} = 15 \ \frac{\text{grams}}{\text{cookie}}$.
Similarly,
$50 \ \frac{\text{grams}}{\text{egg}} \times H_e \ \frac{\text{eggs}}{\text{cookie}} + H_s \ \frac{\text{grams}}{\text{cookie}} = 14 \ \frac{\text{grams}}{\text{cookie}}$.

We use the same iPython code as before, but with the updated information. This returns the following vector:

$$\begin{bmatrix} D_e \\ H_e \\ D_s \\ H_s \end{bmatrix} = \begin{bmatrix} 0.1946 \\ 0.0822 \\ 5.3571 \\ 10.000 \end{bmatrix}$$

With 5 equations the sum of squared errors is $||e||^2 = 0.0029$ and with 7 equations $||e||^2 = 0.0339$. We notice that with 7 equations the error increased. This is not surprising since we added more data points that our model is unable to predict perfectly. Thus we have more terms in the sum and more error. What is more interesting is that for this problem even when we calculate the average error for a given data point

$$\text{5 equations:} \qquad ||e||^2 = 0.0029 \rightarrow ||e|| = 0.0535 \rightarrow \frac{||e||}{5} = 0.0107 \tag{11}$$

$$\text{7 equations:} \qquad ||e||^2 = 0.0339 \rightarrow ||e|| = 0.1841 \rightarrow \frac{||e||}{N} = 0.0263. \tag{12}$$

it is larger for the 7 equation model. What this means is that our new observations are either noisier than the previous observations or they disagree with the model suggested by the first 5 data points. This is not necessarily a bad thing. If we know that the new data points are noisier than the old data points there are ways to adjust the least squares formula to compensate for this (outside the scope of this course). If we know that the new data points are actually quite accurate, this discrepancy could suggest that our linear model is naive and we have to consider more complicated models. Perhaps our previous model deceived us into thinking we had a good fit when we really were just not looking at

the right data. Since we are trying to build a model that estimates the real world, we don't ever want to throw out good data just to make our model fit better.

Rubric Note: Give yourself full credit if you calculated the magnitude of the error vector for Parts B and C. Don't take away points from yourself for your explanation to "Are the new values more accurate?" That question was meant to get you thinking about why the Least Squares technique would output certain results.

7. 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?

Solution:

I worked on this homework with...

I first worked by myself for 2 hours, but got stuck on problem 5, so I went to office hours on...

Then I went to homework party for a few hours, where I finished the homework.