EECS 16A Designing Information Devices and Systems I
Summer 2017 D. Aranki, F. Maksimovic, V. Swamy Homework 7

This homework is due on Sunday, August 6, 2017, at 23:59.
Self-grades are due on Monday, August 7, 2017, at 23:59.

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

- `hw7.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.

- `hw7.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: Change of Basis

All calculations in this problem are intended to be done by hand, but you can use a computer to check your work.

(a) Consider two bases for $\mathbb{R}^2$, A and B, represented by the columns of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

Suppose $\vec{x}_A$ represents the coordinates of a vector $\vec{x}$ in basis A and $\vec{x}_B$ represents the coordinates of the same vector in basis B. Write the coordinate transformation that converts $\vec{x}_A$ to $\vec{x}_B$. That is, find the matrix $\mathbf{T}$, such that

$$\vec{x}_B = \mathbf{T}\vec{x}_A.$$

(b) Consider two bases for $\mathbb{R}^2$, A and B, represented by the columns of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

Suppose $\vec{x}_A$ represents the coordinates of a vector $\vec{x}$ in basis A and $\vec{x}_B$ represents the coordinates of the same vector in basis B. Write the coordinate transformation that converts $\vec{x}_A$ to $\vec{x}_B$, that is, find the matrix $\mathbf{T}$, such that

$$\vec{x}_B = \mathbf{T}\vec{x}_A.$$

(c) Consider two bases for $\mathbb{R}^3$, A and B, represented by the columns of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively.

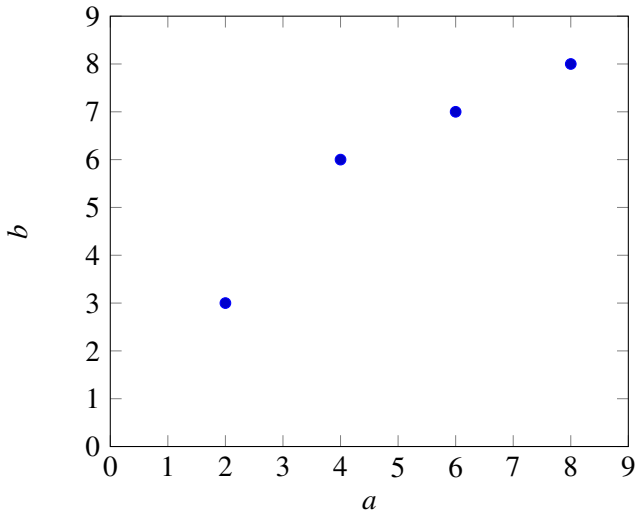
$$\mathbf{A} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & -1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Suppose $\vec{x}_A$ represents the coordinates of a vector $\vec{x}$ in basis A and $\vec{x}_B$ represents the coordinates of the same vector in basis B. Write the coordinate transformation that converts $\vec{x}_A$ to $\vec{x}_B$, that is, find the matrix $\mathbf{T}$, such that

$$\vec{x}_B = \mathbf{T}\vec{x}_A$$

Hint: What do you notice about A and B that will simplify this calculation?

2. Mechanical: Linear Least Squares



a	2	4	6	8
b	3	6	7	8

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

$$b = xa$$

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

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

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 b values as shown in Equation 1. Plot the best fit line along with the data points to examine the quality of the fit. (If you're plotting by hand, it is okay if your plot of $b = xa$ is approximate.)

(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

$$b = x_1 a + x_2.$$

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.

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

Compute the squared error between your model's prediction and the actual 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.)

3. Inner Products

For each of the following functions, show whether it defines an inner product on the given vector space. If not, give a counterexample.

(a) For $\mathbb{R}^2$:

$$\langle \vec{p}, \vec{q} \rangle = \vec{p}^T \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \vec{q}$$

(b) For $\mathbb{R}^2$:

$$\langle \vec{p}, \vec{q} \rangle = \vec{p}^T \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix} \vec{q}$$

(c) For $\mathbb{R}^2$:

$$\langle \vec{p}, \vec{q} \rangle = \vec{p}^T \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \vec{q}$$

(d) For $\mathbb{P}_2$ (the vector space containing all polynomials of up to degree 2):

$$\langle p(x), q(x) \rangle = p(-1)q(-1) + p(0)q(0) + p(1)q(1)$$

(e) For $\mathbb{P}_3$ (the vector space containing all polynomials of up to degree 3):

$$\langle p(x), q(x) \rangle = p(-1)q(-1) + p(0)q(0) + p(1)q(1)$$

(f) For $\mathbb{P}_1$ (the vector space containing all polynomials of up to degree 1):

$$\langle p(x), q(x) \rangle = \int_0^1 p(x)q(x)dx$$

In other words, if $p(x) = ax + b$ and $q(x) = cx + d$, then $\langle p(x), q(x) \rangle = \frac{ac}{3} + \frac{ad}{2} + \frac{bc}{2} + bd$.

4. Finding Signals in Noise

Disclaimer: This problem looks long. However, almost all of the parts involve only running the provided IPython code and commenting on its output.

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 that there is a transmitter sending a known signal $\vec{s}_1$, which is periodic with 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 $\vec{s}_1$, so that $\vec{s}_1$ has a norm of 1. That is, the vector $\vec{s}_1$ looks like:

$$\vec{s}_1 = \frac{1}{\sqrt{n}} \begin{bmatrix} +1 & -1 & -1 & +1 & -1 & \cdots \end{bmatrix}^T,$$

where the ± 1 entries are chosen randomly.

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 $\vec{s}_1$ 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 $\vec{s}_1$, so $\langle \vec{s}_1, \vec{s}_1 \rangle = 1$.

(b) Suppose we receive a signal $\vec{y}$, which is $\vec{s}_1$ 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 largest 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 $\vec{s}_1$ (i.e. 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 that we receive 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 $\vec{s}_1$.

Run the provided IPython code to compute $\langle \vec{s}_1, \vec{n} \rangle$. Run this a few times for different random choices of $\vec{s}_1, n$. 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 that 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 is 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 that 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 $\vec{s}_1$? What about $\vec{s}_2$? (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 that 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).

5. OMP Practice

(a) Suppose we have a vector $\vec{x} \in \mathbb{R}^4$. We take 3 measurements of it, $b_1 = \vec{m}_1^T \vec{x} = 4$, $b_2 = \vec{m}_2^T \vec{x} = 6$, and $b_3 = \vec{m}_3^T \vec{x} = 3$, where $\vec{m}_1$, $\vec{m}_2$ and $\vec{m}_3$ are some measurement vectors. In the general case when there are 5 unknowns in $\vec{x}$ and we only have 3 measurements, it is not possible to solve for $\vec{x}$. Furthermore, there could be noise in the measurements. However in this case, we are given that $\vec{x}$ is sparse and only has 2 non-zero entries. In particular,

$$\mathbf{M}\vec{x} \approx \vec{b}$$

$$\begin{bmatrix} - & \vec{m}_1^T & - \\ - & \vec{m}_2^T & - \\ - & \vec{m}_3^T & - \end{bmatrix} \vec{x} \approx \vec{b}$$

$$\begin{bmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \approx \begin{bmatrix} 4 \\ 6 \\ 3 \end{bmatrix}$$

where exactly 2 of x_1 to x_4 are non-zero. Use Orthogonal Matching Pursuit to estimate x_1 to x_4.

(b) We know that OMP works only when the vector $\vec{x}$ is sparse, which means that it has very few non-zero entries. What if $\vec{x}$ is not sparse in the standard basis but is only sparse in a different basis? What we can do is to change to the basis where $\vec{x}$ is sparse, run OMP in that basis, and transform the result back into the standard basis. Suppose we have a $m \times n$ measurement matrix $\mathbf{M}$ and a vector of measurements $\vec{b} \in \mathbb{R}^m$ where $\mathbf{M}\vec{x} = \vec{b}$ and we want to find $\vec{x} \in \mathbb{R}^n$. The basis that $\vec{x}$ is sparse in is defined by basis vectors $\vec{a}_1 \cdots \vec{a}_n$, and we define:

$$\mathbf{A} = \begin{bmatrix} | & & | \\ \vec{a}_1 & \cdots & \vec{a}_n \\ | & & | \end{bmatrix}$$

such that $\vec{x} = \mathbf{A}\vec{x}'$ and that $\vec{x}'$ is sparse.

Suppose that we have an OMP function that will compute $\vec{x}'$ given $\mathbf{M}'$ and $\vec{b}'$ **only when** $\vec{x}'$ **is sparse**: $\vec{x}' = \text{OMP}(\mathbf{M}', \vec{b}')$. Assuming that the change of basis does not significantly affect the orthogonality of vectors, describe how you would compute $\vec{x}$ using the function OMP.

6. Audio File Matching

Lots of different quantities we interact with every day can be expressed as vectors. For example, an audio clip can be thought of as a vector. The series of numbers in the clip determines the sounds we hear. An audio segment or a sound wave is a continuous function of time, but this can be sampled at regular intervals to make a discrete sequence of numbers that can be represented as a vector.

This problem explores using inner products to measure similarity. The ideas here are similar to the themes of Locationing and GPS.

Let us consider a very simplified model for an audio signal, one that is just composed of two tones. One is represented by -1 and the other by $+1$. A vector of length N makes up the audio file.

(a) Say we want to compare two audio files of the same length N to decide how similar they are. First, consider two vectors that are exactly identical $\vec{x}_1 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ and $\vec{x}_2 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$. What is the dot product of these two vectors? What if $\vec{x}_1 = \begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^T$ and $\vec{x}_2 = \begin{bmatrix}1 & -1 & 1 & -1 & \cdots & 1 & -1\end{bmatrix}^T$ (where the length of the vector is an even number)? Can you come up with an idea to compare two general vectors of length N now?

(b) Next suppose we want to find a short audio clip in a longer one. We might want to do this for an application like *Shazam*, which is able to identify a song from a signature tune. Consider the vector of length 8, $\vec{x} = \begin{bmatrix}-1 & 1 & 1 & -1 & 1 & 1 & -1 & 1\end{bmatrix}^T$. Let us label the elements of $\vec{x}$ so that $\vec{x} = \begin{bmatrix}x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8\end{bmatrix}^T$. We want to find the short segment $\vec{y} = \begin{bmatrix}1 & 1 & -1\end{bmatrix}^T$ in the longer vector, i.e. we want to find i, such that the sequence represented by $\begin{bmatrix}x_i & x_{i+1} & x_{i+2}\end{bmatrix}^T$ is the closest to $\vec{y}$. How can we find this? Applying the same technique, what i gives the best match for $\vec{y} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$?

(c) Now suppose our vector is represented using integers and not just by 1 and -1. Say we want to locate the sequence closest to $\vec{y} = \begin{bmatrix}1 & 2 & 3\end{bmatrix}^T$ in $\vec{x} = \begin{bmatrix}1 & 2 & 3 & 4 & 5 & 6 & 7 & 8\end{bmatrix}^T$. What happens if you apply the technique of part (b)? How would you modify this technique for the problem here?

(d) Answer part (d) in the provided IPython notebook.

(e) Answer part (e) in the provided IPython notebook.

7. Labeling Patients Using Gene Expression Data

Least squares techniques are useful for many different kinds of prediction problems. The core ideas we learned in class have been extensively further developed. 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, family history, is often used to understand the likelihood that a patient might develop certain common diseases such as diabetes. Gene expression profiles can be read using DNA microarray technology, which uses tissue samples from a patient. This data, along with the patient specific characteristics above, can be combined into a vector to get a set of features that describe each patient.

Many scientific studies look at models in mice to understand how gene expression relates to diabetes. Previous 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 information about the age and weight of the mouse and additionally have access to data about whether the genes tomosin2, ts1 and chn1 (a third gene) were expressed or not. The gene expression data is

captured using vectors that are $+1$ if the gene is expressed and -1 if the gene is not expressed. Similarly, whether or not a mouse has diabetes is also captured using a $+1,-1$ vector, where $+1$ indicates that the mouse has diabetes. Using this data we would like to develop a linear model that predicts whether or not a mouse will have diabetes.

$$\alpha_1(\text{age}) + \alpha_2(\text{weight}) + \alpha_3(\text{tomosin2}) + \alpha_4(\text{ts1}) + \alpha_5(\text{chn1})$$

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

(a) In problems such as this, it is common to use some *training* data to generate a model. Turns out, a good heuristic for this can be developed using a least squares technique. Set up a linear model for the problem in a format we have used for least squares problems $\mathbf{A}\vec{x} = \vec{b}$. Here, $\vec{b}$ will be a vector with $+1,-1$ entries. The α_i's are your unknowns.

(b) Using the (fake) *training* data `diabetes_train.npy`, generate the linear model using the least squares technique, i.e. find the unknown model parameters for the given data set. Include the unknown parameter values in the writeup of your homework. Use the provided IPython notebook.

(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 writeup.

8. Image Analysis

Applications in medical imaging often require an analysis of images based on the pixels of the image. 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 the locations and thus the number of cells. Alternatively, 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, so you want to find the equation of the ellipse that bounds the cell relative to a given coordinate system that is represented by 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 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 a quadratic 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 a quadratic 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 set of matrix equations to do this, i.e. an equation of the form $\mathbf{A}\vec{x} = \vec{b} + \vec{e}$, where $\vec{b}$ represents your observations and $\vec{e}$ represents the unknown errors.

(b) How can you find the equation of an ellipse that surrounds the cell? Provide a setup and formulate a set of matrix equations to do this as above.

(c) In the IPython notebook, write a short program to fit a circle using these points. If you model your system of equations as $\mathbf{A}\vec{x} = \vec{b} + \vec{e}$, where $\vec{e}$ is the error vector and the number of data points is N, what is $\frac{\|\vec{e}\|}{N}$? Plot your points and the best fit circle in IPython.

(d) Write a short program in IPython to fit an ellipse using these points. If you model your system of equations as $\mathbf{A}\vec{x} = \vec{b} + \vec{e}$, where $\vec{e}$ is the error vector and the number of data points is N, what is $\frac{\|\vec{e}\|}{N}$? 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?

9. 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 these 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 accurately 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 non-linear 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

$$p = 1 - 0.95^{e^{(R-26.1931)}}, \tag{3}$$

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:

$$\begin{aligned} R =& x_1 \cdot \ln\Big(\text{AGE (years)}\Big) + x_2 \cdot \ln\Big(\text{TC (mg/dL)}\Big) + \\ & x_3 \cdot \ln\Big(\text{HDL (mg/dL)}\Big) + x_4 \cdot \ln\Big(\text{SBP (mm Hg)}\Big) + \\ & x_5 \cdot \Big(\text{DIA (binary)}\Big) + x_6 \cdot \Big(\text{SMK (binary)}\Big) \end{aligned} \tag{4}$$

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

$$b^k = x_1 A_1^k + x_2 A_2^k + x_3 A_3^k + x_4 A_4^k + x_5 A_5^k + x_6 A_6^k \tag{5}$$

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 3 in the form of Equation 5, 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) (`https://www.cdc.gov/nchs/nhanes/index.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

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

where $\mathbf{A}$ is a tall matrix and $\vec{b}$ is a tall vector. What form should $\vec{b}$ and $\mathbf{A}$ 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 the IPython notebook is for 91 patients. Construct $\vec{b}$ and $\mathbf{A}$ using the loaded data. What are the dimensions of $\vec{b}$ and A?

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

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

(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=40 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 $\vec{x}$ slightly and re-plotting. Does it look like the fit is getting better or worse? Is the squared error increasing or decreasing?

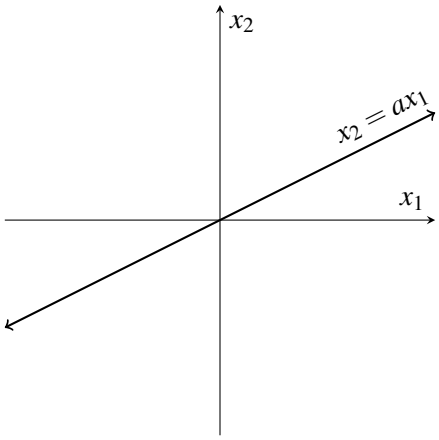
(h) **PRACTICE:** Transform b and $\hat{b}$ back into the form of p and transform A_1, A_2, A_3 back into the form of AGE, TC, and HDL and re-plot. What do you see?

(i) **PRACTICE:** 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.

10. Perceptron Valley

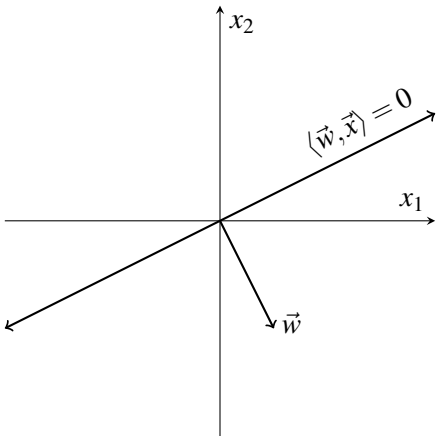
(a) Consider the following line in $\mathbb{R}^2$:



We know a line through the origin can be written using the equation $x_2 = ax_1$, where a represents the slope, but this is not the only equation that defines a line.

Let $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Write an equation that defines the same line as $x_2 = ax_1$, but in the form $\langle \vec{w}, \vec{x} \rangle = 0$. What is $\vec{w}$ in terms of a? Is $\vec{w}$ unique?

(b) The meaning of the equation $\langle \vec{w}, \vec{x} \rangle = 0$ is that the line we want is formed by all possible vectors $\vec{x}$ that are orthogonal to a particular vector $\vec{w}$.



In this way, the vector $\vec{w}$ defines the line. If a vector $\vec{x}$ does not lie on the line, then it will have a non-zero inner product with $\vec{w}$. Some choices of $\vec{x}$ will have a positive inner product with $\vec{w}$, and some will have a negative inner product.

Suppose $\vec{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

i. Draw and label a 2-D Cartesian plane with x_1 on the horizontal axis and x_2 on the vertical axis.

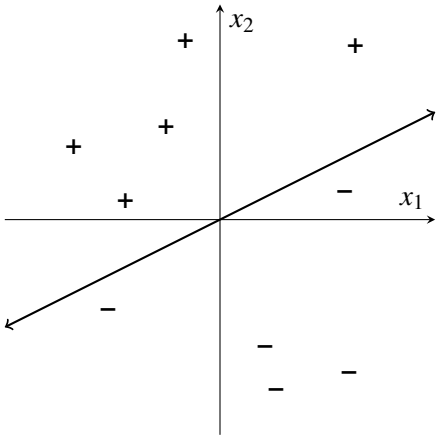
ii. Draw and label the vector $\vec{w}$.

iii. Draw the line defined by $\langle \vec{w}, \vec{x} \rangle = 0$.

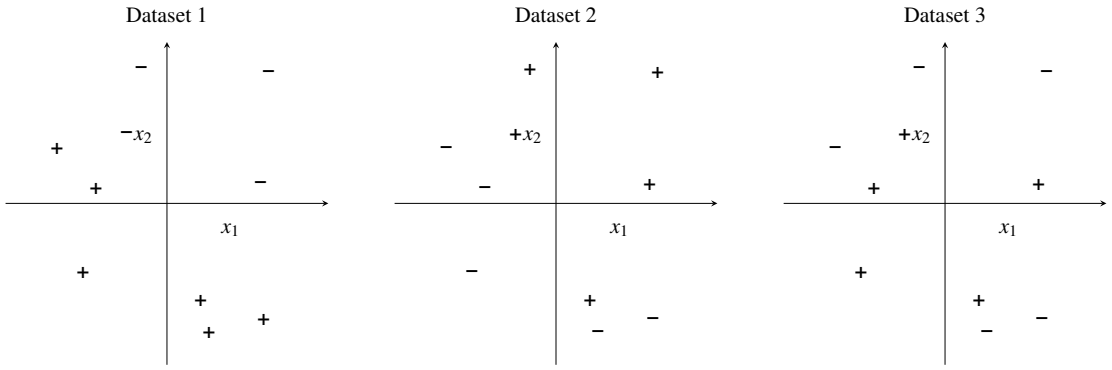
iv. Label, with a '+', the region of the plane containing all vectors $\vec{x}$ that make a positive inner product with $\vec{w}$. In other words, label the set $\left\{ \vec{x} \in \mathbb{R}^2 | \, \langle \vec{w}, \vec{x} \rangle > 0 \right\}$ with a '+' sign.

v. Label, with a '–', the region of the plane containing all vectors $\vec{x}$ that make a negative inner product with $\vec{w}$. In other words, label the set $\left\{ \vec{x} \in \mathbb{R}^2 | \, \langle \vec{w}, \vec{x} \rangle < 0 \right\}$ with a '–' sign.

(c) Now suppose we collect some data from an experiment where each data point consists of two numbers x_1 and x_2 and a *label* of either '+' or '–'. We plot our data points on the axis below.



If we can draw a line that separates all the '+' data points from the '–' data points, then we say our dataset is *linearly separable* and this line is called a *separating hyperplane*. Just like a plane is a 3-D version of a 2-D line, a hyperplane generalizes a plane to any number of dimensions. For this problem, we will stay in $\mathbb{R}^2$ for simplicity and we will also only consider hyperplanes containing the origin. For each of the following datasets, say whether it is linearly separable or not. If it is, give an example of a vector $\vec{w}$ that defines a separating hyperplane.



(d) The problem of finding a separating hyperplane for a given dataset is a special case of a well-studied problem in supervised machine learning called *binary classification*. In the supervised binary classification problem, we are given a *training set* consisting of (vector, label) pairs, where the label $y \in \{+1, -1\}$. The vectors are often called *feature vectors* and represent quantities extracted from an input. For example, if the input is an image of a food, we could use image processing and computer vision techniques to extract features (e.g. shape, color, etc.) and concatenate these features into a feature vector. The label for each feature vector might be assigned according to whether or not the image is of a hot dog: $+1$ for "hot dog" and -1 for "not hot dog".
We can use the training set to *train* a linear classification algorithm, which tries its best to find a separating hyperplane. If it successfully finds one, we can then take a *new, unlabeled* input, extract its feature vector, and predict its label based on which side of the hyperplane it's on. In practice, the training set is often not linearly separable, so the algorithm will have non-zero *training error*.
The **Perceptron Learning Algorithm** is one of the oldest and simplest linear classification algorithms. It starts with an initial guess of a hyperplane defined by $\vec{w}_0$, often called the initial *weights*. Then it randomly samples a (feature vector, label) pair from the training set $(\vec{x}_i, y_i)$. If $\vec{w}_0$ correctly predicts y_i, then nothing happens. However, if it predicts the wrong label, then the weights are updated as

$$\vec{w}_1 \leftarrow \vec{w}_0 + y_i \vec{x}_i$$

In general in the j^{th} iteration if $\vec{w}_j$ incorrectly predicts y_i, the following happens

$$\vec{w}_{j+1} \leftarrow \vec{w}_j + y_i \vec{x}_i$$

Suppose that $y_i = +1$ but $\text{sign}\left(\left\langle \vec{w}_j, \vec{x}_i \right\rangle\right) = -1$. In other words, the current weights predict incorrectly because the inner product is too negative. Assuming that $\vec{x}_i \neq \vec{0}$, prove that the update makes the inner product less negative/more positive, that is, **prove that**

$$\text{for } y_i = +1 \text{ and } \text{sign}\left(\left\langle \vec{w}_j, \vec{x}_i \right\rangle\right) = -1,$$
$$\left\langle \vec{w}_{j+1}, \vec{x}_i \right\rangle > \left\langle \vec{w}_j, \vec{x}_i \right\rangle.$$

When $y_i = -1$ but $\text{sign}\left(\left\langle \vec{w}_j, \vec{x}_i \right\rangle\right) = +1$, the opposite occurs. **Prove that**

$$\text{for } y_i = +1 \text{ and } \text{sign}\left(\left\langle \vec{w}_j, \vec{x}_i \right\rangle\right) = -1, \left\langle \vec{w}_{j+1}, \vec{x}_i \right\rangle > \left\langle \vec{w}_j, \vec{x}_i \right\rangle.$$

When $y_i = -1$ but $\text{sign}\left(\left\langle \vec{w}_j, \vec{x}_i \right\rangle\right) = +1$, the opposite occurs. Show that

$$\text{for } y_i = -1 \text{ and } \text{sign}\left(\left\langle \vec{w}_j, \vec{x}_i \right\rangle\right) = +1, \left\langle \vec{w}_{j+1}, \vec{x}_i \right\rangle < \left\langle \vec{w}_j, \vec{x}_i \right\rangle.$$

In this way, the update pushes the hyperplane in the right direction. This is by no means a proof that the PLA always finds a separating hyperplane if there is one, but it gives some insight into how it works.

(e) The PLA algorithm repeats the process of updating the hyperplane equation for a specified number of iterations or until the training error is low enough. In this class, we will use a maximum iteration stopping criterion. You will learn about training error stopping criterion in CS189. The algorithm is concisely defined below.

The Perceptron Learning Algorithm (PLA):

Inputs:

- Training data $\{(\vec{x}_i, y_i)\}_{i=1}^n$, where $\vec{x}_i \in \mathbb{R}^d$, $y_i \in \{+1, -1\}$ for $i = 1, 2, \ldots, n$
- Initial weights $\vec{w}_0 \in \mathbb{R}^d$
- A maximum iteration number J

Outputs:

- Trained weights $\vec{w}_J$

procedure PLA($\{(\vec{x}_i, y_i)\}_{i=1}^n$, $\vec{w}_0$, J)
    $j \leftarrow 0$ (iteration number)
    while $j < J$ **do**
        $i \leftarrow$ `RandInt`(n) (random integer in range $[1, n]$)
        if sign $\left(\left\langle \vec{w}_j, \vec{x}_i \right\rangle\right) \neq y_i$ **then**
            $\vec{w}_{j+1} \leftarrow \vec{w}_j + y_i \vec{x}_i$
        else
            $\vec{w}_{j+1} \leftarrow \vec{w}_j$
        end if
        $j \leftarrow j + 1$
    end while
end procedure

In the IPython Notebook, implement the PLA by filling in the missing lines of code. Watch it learn in action and feel free to play around with the numbers! Remember that the PLA is a random algorithm, so the results will be different every time. Fun fact: The PLA can be viewed as a single neuron of an Artificial Neural Network using sign as its non-linear activation function!

11. Constrained Least-Squares Optimization

In this problem, you'll go through a process of guided discovery to solve the following optimization problem:

Consider a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, of full column rank, where $M > N$. Determine a unit vector $\vec{x}$ that minimizes $\|\mathbf{A}\vec{x}\|$, where $\|\cdot\|$ denotes the 2-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:

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

This task may *seem* like solving a standard least-squares problem $\mathbf{A}\vec{x} = \vec{b}$, where $\vec{b} = \vec{0}$, but it isn't. An important distinction is that in our problem, $\vec{x} = \vec{0}$ is *not* a valid solution, because the zero vector does not have unit length. Our optimization problem is a least squares problem with a constraint—hence the term *Constrained Least-Squares Optimization*. The constraint is that the vector $\vec{x}$ must lie on the unit sphere in $\mathbb{R}^N$. You'll tackle this problem in a methodical, step-by-step fashion.

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 and indexed in an ascending fashion—that is,

$$\lambda_1 \leq \cdots \leq \lambda_N.$$

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 $0 < \lambda_1$.

(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

$$\mathbf{A}^T\mathbf{A}\,\vec{v}_k = \lambda_k\,\vec{v}_k \tag{6}$$

and

$$\vec{v}_\ell^{\,T}\mathbf{A}^T\mathbf{A} = \lambda_\ell\,\vec{v}_\ell^{\,T}. \tag{7}$$

Premultiply Equation 6 with $\vec{v}_\ell^{\,T}$, postmultiply Equation 7 with $\vec{v}_k$, compare the two, and explain how one may then infer that $\vec{v}_k$ and $\vec{v}_\ell$ are orthogonal.

(c) Since the N eigenvectors of $\mathbf{A}^T\mathbf{A}$ are mutually orthogonal—and each has unit length—they 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 $\vec{v}_1, \ldots, \vec{v}_N$.

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

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

(d) Now you're well-positioned to tackle the grand challenge of this problem—determine the unit vector $\vec{x}$ that minimizes $\|\mathbf{A}\vec{x}\|$.

Note that the task is the same as finding a unit vector $\vec{x}$ that minimizes $\|\mathbf{A}\vec{x}\|^2$.

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 $\hat{\vec{x}}$ such that $\|\mathbf{A}\vec{x}\|^2$ is minimized. You may *not* use any tool from calculus to solve this problem—so avoid differentiation of any flavor.

For the optimal vector $\hat{\vec{x}}$ that you determine—that is, the vector

$$\hat{\vec{x}} = \text{argmin}_{\vec{x}}\,\|\mathbf{A}\vec{x}\|^2 \quad \text{subject to the constraint} \quad \|\vec{x}\|^2 = 1,$$

determine a simple, closed-form expression for the minimum value

$$\min_{\|\vec{x}\|=1} \|\mathbf{A}\vec{x}\| = \left\|\mathbf{A}\hat{\vec{x}}\right\|.$$

12. Homework Process

(a) 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?

Working in groups of 3-5 will earn you credit for your participation grade.

(b) Copy the following statement into your homework submission.

> I, [insert name here], affirm that all of my solutions are entirely my work and that I have properly credited and acknowledged any sources or work that I have consulted.