EECS 16A Designing Information Devices and Systems I
Spring 2018 Homework 13

This homework is due April 30, 2018, at 23:59.
Self-grades are due May 3, 2018, at 23:59.

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

- `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.

- `hw13.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. 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 $0 < \lambda_1$.

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 $\mathbf{A}$ is full rank, the numerator is never 0 unless $\vec{v} = \vec{0}$.

(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

$$\mathbf{A}^T\mathbf{A}\vec{v}_k = \lambda_k\vec{v}_k \tag{1}$$

and

$$\vec{v}_\ell^T\mathbf{A}^T\mathbf{A} = \lambda_\ell\vec{v}_\ell^T. \tag{2}$$

Premultiply Equation 1 with $\vec{v}_\ell^T$, postmultiply Equation 2 with $\vec{v}_k$, compare the two, and explain how one may then infer that $\vec{v}_k$ and $\vec{v}_\ell$ are orthogonal.

Solution:

Following the hint:

$$\begin{aligned}\vec{v}_\ell^T\mathbf{A}^T\mathbf{A}\vec{v}_k &= \vec{v}_\ell^T\lambda_k\vec{v}_k\\ \vec{v}_\ell^T\mathbf{A}^T\mathbf{A}\vec{v}_k &= \lambda_\ell\vec{v}_\ell^T\vec{v}_k\end{aligned}$$

We see the two expressions on the left are equal, so 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) 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 $\vec{v}_1, \ldots, \vec{v}_N$.

Solution:

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

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

$$\alpha_i = \langle\vec{x}, \vec{v}_i\rangle = \vec{x}^T\vec{v}_i$$

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

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

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

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

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

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

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:

$$\begin{aligned}\vec{x}^T\mathbf{A}^T\mathbf{A}\vec{x} &= \vec{x}^T \sum_{n=1}^{N} \alpha_n \lambda_n \vec{v}_n \\ &= \sum_{n=1}^{N} \alpha_n^2 \lambda_n\end{aligned}$$

To minimize, we pick the entire weight of α of the smallest λ, i.e. we pick $\vec{x}$ to be the eigenvector $\vec{v}_i$ with the smallest eigenvalue, and the minimum value is λ_i.

2. 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.

Solution:

The unknowns that we want to find are α_i, where $\{i = 1,\ldots,5\}$. The data we are given has the age, weight, and expressions of the genes tomosin2, ts1, and chn1, so the matrix $\mathbf{A}$ would be the matrix with each row containing the data of each mouse. The vector $\vec{b}$ is a column vector indicating if the mouse has diabetes or not. The vector $\vec{x}$ is the vector of unknowns $\vec{x} = \begin{bmatrix}\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5\end{bmatrix}^T$. Hence, the setup is:

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

(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.

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

$$\vec{x} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} -0.15646169 \\ 0.09239418 \\ 0.48053974 \\ -0.5847018 \\ -0.35350734 \end{bmatrix}$$

(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 writeup.

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 is exactly the values given to us in `diabetes_test.npy`.

3. How Much Is Too Much?

When discussing circuits in this course, we only talked about resistor I-V curves. There are many other two terminal devices that can be found in nature that do not have linear I-V relations. Instead, I is some general function of V, that is $I = f(V)$. Often times, the function describing the I-V relationship is not known beforehand. Furthermore, noise is present in every measurement. The function f is assumed to be a polynomial, and the parameters of f (the coefficients for every power of V) are computed using least squares.

Throughout this problem, we are provided with $\vec{x}$, a set of voltage measurements, and $\vec{y}$, a set of current measurements.

(a) Let's first try to model a resistor I-V curve. Run the code in the attached IPython notebook. Comment on the fit with different degree polynomials.

Solution:

The higher the degree of the polynomial, the more are we fitting the noise in the signal. Intuitively, we want to use a degree 1 polynomial (a line) to model the roughly linear data. As we increase the degree of the polynomial, the best fit polynomial starts to fluctuate, which means that it starts to fit the noise. Since we don't want to fit the noise, the degree of the polynomial we choose should not be too large.

(b) In the attached IPython Notebook, fill out the code for the cost function. The cost function returns the mean squared error, or $\|\vec{I} - \mathbf{A}\vec{f}\|$, where $\mathbf{A}$ is the appropriately sized matrix containing powers of V. Then plot the cost of various degree polynomials fitting to the measured I-V points for a resistor. Comment on the shape of the Cost vs. Degree graph.

Solution:

We observe that the cost decreases as the degree of the polynomial increases. This is expected because as we increase the degree of the polynomial, we start to fit all of the data points in the data set, including the noise. This means that the best fit polynomial is "closer" to the data points, so the cost decreases.

(c) One issue with testing on the same data that we perform least squares on is that we end up fitting to noise. For this reason, the data set is generally divided into two sets of data. One set is the training set, which is used to train the model. In this case, we will use least squares on the training set to calculate the parameters to our polynomial approximation. The other data set is the test set, where we test our model. This is the data set we will use to compute the cost for a given degree polynomial. In the attached IPython notebook, fill in the code for the `improvedCost` function. Then plot the Cost vs. Degree graph as before and comment on the results.

Solution:

We observe that the cost is at its minimum when the degree equals 1. This is because we were previously fitting the noise that existed in the training set, but the noise is now different in the test set. This means that the best fit polynomial, the degree 1 polynomial, that did not fit the noise in the training set will now be the best fit for the test set since it does not contain the noise in the training set. This graph also shows that the optimal degree of our best fit polynomial is 1.

(d) So far, we've only used least squares to fit to a resistor. Let's repeat this procedure for a non-linear device. In the attached IPython notebook, plot the Cost vs. Degree curve for this new device. From this graph, what order polynomial should we use to approximate f?

Solution:

From the graph, we observe that the cost is at its minimum when the degree equals 5. Therefore, we should use a fifth order polynomial to approximate f.

Also notice that the cost is extremely high if we use a polynomial of degree 1 or 2.

(e) Let's repeat this procedure for one final device. Use the attached IPython notebook to plot the I-V data points for this device. From the I-V curve of the device, what type of function if f? Is it a polynomial? Plot the Cost vs. Degree graph. Explain the shape of the Cost vs. Degree graph given what you know about f.

Solution:

f is an exponential function, which is not a polynomial but an infinite power series. The cost decreases monotically as the degree increases because the exponential function is a sum of x^i from $i = 0, 1, \ldots$. Therefore, as we increase the degree, our approximation will be closer to the exponential function, so we expect the cost to decrease.

4. 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 4 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.

Solution:

Let $\vec{c}_1$ to $\vec{c}_4$ be the column vectors of $\mathbf{M}$. We first find the column vector in $\mathbf{M}$ that correlates most with $\vec{b}$:

$$\begin{bmatrix} \langle \vec{c}_1, \vec{b} \rangle & \langle \vec{c}_2, \vec{b} \rangle & \langle \vec{c}_3, \vec{b} \rangle & \langle \vec{c}_4, \vec{b} \rangle \end{bmatrix} = \begin{bmatrix} 10 & 7 & -3 & -1 \end{bmatrix}$$

Thus, $\vec{c}_1$ is the best matching vector, and we compute

$$\vec{b}' = \vec{b} - \vec{c}_1 \frac{\langle \vec{b}, \vec{c}_1 \rangle}{\langle \vec{c}_1, \vec{c}_1 \rangle} = \begin{bmatrix} -1 \\ 1 \\ 3 \end{bmatrix}$$

which is the subtraction of the projection of $\vec{b}$ onto $\vec{c}_1$ from $\vec{b}$. Then we find the largest correlation again:

$$\begin{bmatrix} \langle \vec{c}_1, \vec{b}' \rangle & \langle \vec{c}_2, \vec{b}' \rangle & \langle \vec{c}_3, \vec{b}' \rangle & \langle \vec{c}_4, \vec{b}' \rangle \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 & 4 \end{bmatrix}$$

Thus, we know that $\vec{c}_1$ and $\vec{c}_4$ contribute most to $\vec{b}$. However no linear combination of $\vec{c}_1$ and $\vec{c}_4$ can form $\vec{b}$ because of noise in the measurements. Thus, we need to find the least squares solution. Let $\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$, and the least squares formula gives:

$$\begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \vec{b} = \begin{bmatrix} 6\frac{1}{3} \\ 2\frac{2}{3} \end{bmatrix}$$

Thus, $\vec{x} \approx \begin{bmatrix} 6\frac{1}{3} \\ 0 \\ 0 \\ 2\frac{2}{3} \end{bmatrix}$.

(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.

Solution:

Since

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

We get:

$$\vec{x}' = \text{OMP}(\mathbf{MA}, \vec{b})$$

And the original $\vec{x}$ is thus:

$$\vec{x} = \mathbf{A}\vec{x}' = \mathbf{A}\ (\text{OMP}(\mathbf{MA}, \vec{b}))$$

Out-of-scope remark: In certain cases (choice of $\mathbf{M}$ and $\mathbf{A}$), the assumption that "the change of basis does not significantly affect the orthogonality of vectors" can be formalized. We want it to be the case that if columns of $\mathbf{M}$ are roughly orthogonal, then so are columns of $\mathbf{MA}$. It turns out that if the measurement matrix $\mathbf{M}$ is chosen randomly (similarly to what you have seen in precious HWs) and the basis $\mathbf{A}$ is orthogonal, then not only will the columns of $\mathbf{M}$ be roughly orthogonal but also the columns of $\mathbf{MA}$ will be roughly orthogonal. Therefore, our assumption is justified in some cases. (As an exercise, it is possible to come up with "bad" choices of $\mathbf{M}$ and $\mathbf{A}$ for which the assumption strongly fails).

5. Sparse Imaging

Recall the imaging lab where we projected masks on an object to scan it to our computer using a single pixel measurement device, that is, a photoresistor. In that lab, we were scanning a 30×40 image having 1200 pixels. In order to recover the image, we took exactly 1200 measurements because we wanted our 'measurement matrix' to be invertible.

However, we saw that an iterative algorithm that does "matching and peeling" can enable reconstruction of a sparse vector while reducing the number of samples that need to be taken from it. In the case of imaging, the idea of sparsity corresponds to most parts of the image being black with only a small number of light pixels. In these cases, we can reduce the overal number of samples necessary. This would reduce the time required for scanning the image. (This is a real-world concern for things like MRI where people have to stay still while being imaged.)

In this problem, we have a 2D image I of size 91×120 pixels for a total of 10920 pixels. The image is made up of mostly black pixels except for 476 of them that are white.

Although the imaging illumination masks we used in the lab consisted of zeros and ones, in this question, we are going to have masks with real values — i.e. the light intensity is going to vary in a controlled way. Say that we have an imaging mask M_0 of size 91×120. The measurements using the solar cell using this imaging mask can be represented as follows.

First, let us vectorize our image to $\vec{i}$ which is a length 10920 column vector. Likewise, let us vectorize the mask M_0 to $\vec{m}_0$ which is a length 10920 column vector. Then the measurement using M_0 can be represented as

$$b_0 = \vec{m}_0^T \vec{i}.$$

Say we have a total of M measurements, each taken with a different illumination mask. Then, these measurements can collectively be represented as

$$\vec{b} = \mathbf{A}\vec{i},$$

where $\mathbf{A}$ is an $M \times 10920$ size matrix whose rows contain the vectorized forms of the illumination masks, that is

$$\mathbf{A} = \begin{bmatrix} \vec{m}_1^T \\ \vec{m}_2^T \\ \vdots \\ \vec{m}_M^T \end{bmatrix}.$$

To show that we can reduce the number of samples necessary to recover the sparse image I, we are going to only generate 6500 masks. The columns of $\mathbf{A}$ are going to be approximately uncorrelated with each other. The following question refers to the part of IPython notebook file accompanying this homework related to this question.

(a) In the IPython notebook, we call a function `simulate` that generates masks and the measurements. You can see the masks and the measurements in the IPython notebook file. Complete the function `OMP` that does the OMP algorithm described in lecture.

Solution:

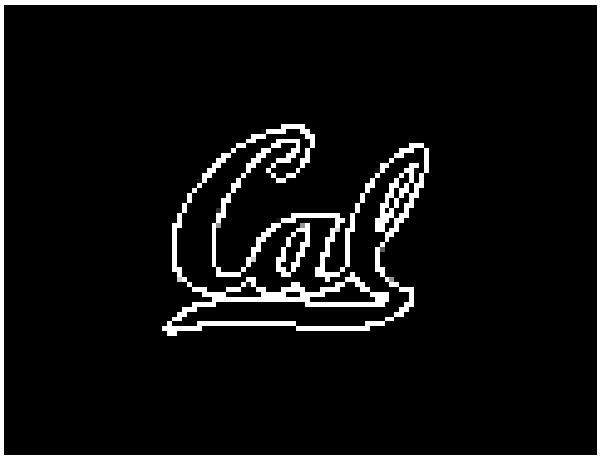
See `sol13.ipynb`.

Remark: Note that this remark is not important for solving this problem; it is about how such measurements could be implemented in our lab setting. When you look at the vector `measurements` you will see that it has zero average value. Likewise, the columns of the matrix containing the masks $\mathbf{A}$ also have zero average value. To satisfy these conditions, they need to have negative values. However, in an imaging system, we cannot project negative light. One way to get around this problem is to find the smallest value of the matrix $\mathbf{A}$ and subtract it from all entries of $\mathbf{A}$ to get the actual illumination masks. This will yield masks with positive values, hence we can project them using our real-world projector. After obtaining the readings using these masks, we can remove their average value from the readings to get measurements as if we had multiplied the image using the matrix $\mathbf{A}$.

(b) Run the code `rec = OMP((height, width), sparsity, measurements, A)` and see the image being correctly reconstructed from a number of samples smaller than the number of pixels of your figure. What is the image?

Solution:

The reconstructed image is the following.



(c) **PRACTICE:** We have supplied code that reads a PNG file containing a sparse image, takes measurements, and performs OMP to recover it. An example input file is also supplied together with the code. Generate an image of size 91×120 pixels of sparsity less than 400 and recover it using OMP with 6500 measurements.

You can answer the following parts of this question in very general terms. Try reducing the number of measurements. Does the algorithm start to fail in recovering your sparse image? Why do you think it fails? Make an image having fewer white pixels. By how much can you reduce the number of measurements that needs to be taken?

Solution:

The answer to this question depends on the size of your input image and the total number of non-zero pixels in it. In order to successfully recover the image, the number of measurements (masks used for imaging) needs to increase as the number of non-zero pixels increases. The reason is, as we have more nonzero pixels, they start to contribute in the measurements (in terms of the example given in the lecture, it is like having more users try to transmit their messages at the same time). In order to not be affected by these contributions, we need the signature of each pixel to be 'more orthogonal' to others. In our setting, this is achieved by taking more measurements with different masks.

6. Mechanical Gram-Schmidt

Use Gram-Schmidt to find a matrix $\mathbf{U}$ whose columns form an orthonormal basis for the column space of $\mathbf{V}$.

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

Show that you get the same resulting vector when you project $\vec{w} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \end{bmatrix}^T$ onto $\mathbf{V}$ and onto $\mathbf{U}$, i.e. show that

$$\mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\vec{w} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\vec{w}.$$

Solution:

We start with the columns of $\mathbf{V}$ as our basis for the column space of $\mathbf{V}$, and we want to find an orthonormal basis for this same space using Gram-Schmidt. For notational convenience, define

$$\mathbf{V} = \begin{bmatrix} | & | & | \\ \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \\ | & | & | \end{bmatrix}$$

We summarize the first few steps of the Gram-Schmidt algorithm as follows:

(a) $\vec{u}_1' = \vec{v}_1; \qquad \vec{u}_1 = \frac{\vec{u}_1'}{\|\vec{u}_1'\|}.$

(b) $\vec{u}_2' = \vec{v}_2 - \langle \vec{v}_2, \vec{u}_1 \rangle \vec{u}_1; \qquad \vec{u}_2 = \frac{\vec{u}_2'}{\|\vec{u}_2'\|}.$

(c) $\vec{u}_3' = \vec{v}_3 - \langle \vec{v}_3, \vec{u}_1 \rangle \vec{u}_1 - \langle \vec{v}_3, \vec{u}_2 \rangle \vec{u}_2; \qquad \vec{u}_3 = \frac{\vec{u}_3'}{\|\vec{u}_3'\|}.$

For the column space of $\mathbf{V}$, this is

(a) $\vec{u}_1' = \vec{v}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$. Since $\vec{u}_1'$ is already normalized, we simply set $\vec{u}_1 = \vec{u}_1'$.

(b)

$$\vec{u}_2' = \vec{v}_2 - \langle \vec{v}_2, \vec{u}_1 \rangle \vec{u}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \vec{u}_2 = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix}$$

(c)

$$\vec{u}_3' = \vec{v}_3 - \langle \vec{v}_3, \vec{u}_1 \rangle \vec{u}_1 - \langle \vec{v}_3, \vec{u}_2 \rangle \vec{u}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \frac{2}{\sqrt{2}} \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \qquad \vec{u}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Thus, the matrix $\mathbf{U}$ is given by

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Note that whatever basis we use for a subspace, when we project a vector onto that subspace, we get the same vector. For example, when we project the vector $\vec{w} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \end{bmatrix}^T$ onto the subspace using the $\mathbf{V}$ basis, we get

$$\begin{aligned} \mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\vec{w} &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 3 \\ 1 & 3 & 5 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{3}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{3}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\vec{w} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \end{aligned}$$

Note however that the projection using the $\mathbf{U}$ basis was much simpler. Since $\mathbf{U}^T\mathbf{U}$ is the identity, we didn't need to do a matrix inversion.

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.