EECS 16B Designing Information Devices and Systems II
Spring 2021 UC Berkeley

Homework 14

This homework is due on Sunday, May 2, 2021, at 11:00PM. Self-grades and HW Resubmission are due on Tuesday, May 4, 2021, at 11:00PM.

1. Reading Lecture Notes

Staying up to date with lectures is an important part of the learning process in this course. Here are links to the notes that you need to read for this week: Note 16, Note 17

(a) Write out an $N \times N$ orthonormal matrix U whose columns represent the DFT basis.

Solution:

$$U = \frac{1}{\sqrt{N}} \begin{bmatrix} e^{j\frac{2\pi(0)(0)}{N}} & e^{j\frac{2\pi(0)(1)}{N}} & e^{j\frac{2\pi(0)(2)}{N}} & \cdots & e^{j\frac{2\pi(0)(N-1)}{N}} \\ e^{j\frac{2\pi(1)(0)}{N}} & e^{j\frac{2\pi(1)(1)}{N}} & e^{j\frac{2\pi(1)(2)}{N}} & \cdots & e^{j\frac{2\pi(N-1)}{N}} \\ e^{j\frac{2\pi(2)(0)}{N}} & e^{j\frac{2\pi(2)(1)}{N}} & e^{j\frac{2\pi(2)(2)}{N}} & \cdots & e^{j\frac{2\pi 2(N-1)}{N}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e^{j\frac{2\pi(N-1)(0)}{N}} & e^{j\frac{2\pi(N-1)}{N}} & e^{j\frac{2\pi 2(N-1)}{N}} & \cdots & e^{j\frac{2\pi(N-1)(N-1)}{N}} \end{bmatrix}$$

In other words, the km^{th} entry $U_{km} = \frac{1}{\sqrt{N}} e^{j\frac{2\pi km}{N}}$

2. Linearization to help classification: discovering logistic regression and how to solve it

You can, in spirit, reduce the problem of linear multi-class classification to that of binary classification by picking vectors that correspond to each of the categories "X" as compared with all the other examples categorized into a hybrid synthetic category of "not-X". This will give rise to vectors corresponding to each category with the winner selected by seeing which one wins. However, we will focus here on the binary problem since that is the conceptual heart of this approach.

As was discussed in lecture, the naive straightforward way of picking the decision boundary (by looking at the mean example of each category and drawing the perpendicular bisector) is not always the best. The included Jupyter Notebook includes synthetic examples that illustrate the different things that can happen so that you can better appreciate the pathway that leads us to discover logistic regression as a natural approach to solve this problem based on the conceptual building blocks that you have already seen.

It is no exageration to say that logistic regression is the default starting method for doing classification in machine learning contexts, the same way that straightforward linear regression is the default starting method for doing regression. A lot of other approaches can be viewed as being built on top of logistic regression. Consequently, getting to logistic regression is a nice ending-point for this part of the 16AB story as pertains to classification.

Let's start by giving things some names. Consider trying to classify a set of measurements $\vec{x}_i$ with given labels ℓ_i. For the binary case of interest here, we will think of the labels as being "+" and "-". For expository convenience, and because we don't want to have to carry it around separately, we will fold our threshold implicitly into the weights by augmenting our given measurements with the constant "1" in the first position of each $\vec{x}_i$. Now, the classification rule becomes simple. We want to learn a vector of weights $\vec{w}$ so that we can deem any point with $\vec{x}_i^\top \vec{w} > 0$ as being a member of the "+" category and anything with $\vec{x}_i^\top \vec{w} < 0$ as being a member of the "-" category.

The way that we will do this, is to do a minimization in the spirit of least squares. Except, instead of necessarily using some sort of squared loss function, we will just consider a generic cost function that can depend on the label and the prediction for the point. For the i-th data point in our training data, we will incur a cost $c(\vec{x}_i^\top \vec{w}, \ell_i)$ for a total cost that we want to minimize as:

$$\arg\min_{\vec{w}} \quad c_{total}(\vec{w}) = \sum_{i=1}^{m} c(\vec{x}_i^\top \vec{w}, \ell_i) \tag{1}$$

Because this can be a nonlinear function, our goal is to solve this iteratively as a sequence of least-squares problems that we know how to solve.

Consider the following algorithm:

1: $\vec{w} = \vec{0}$ ▷ Initialize the weights to $\vec{0}$
2: **while** Not done **do** ▷ Iterate towards solution
3: Compute $\vec{w}^\top \vec{x}_i$ ▷ Generate current estimated labels
4: Compute $\frac{d}{d\vec{w}} c(\vec{w}^\top \vec{x}_i, \ell_i)$ ▷ Generate derivatives with respect to $\vec{w}$ of the cost for update step
5: Compute $\frac{d^2}{d\vec{w}^2} c(\vec{w}^\top \vec{x}_i, \ell_i)$ ▷ Generate second derivatives of the cost for update step
6: $\vec{\delta w} = LeastSquares(\cdot, \cdot)$ ▷ We will derive what to call least squares on
7: $\vec{w} = \vec{w} + \vec{\delta w}$ ▷ Update parameters
8: **end while**
9: Return $\vec{w}$

The key step above is figuring out with what arguments to call LeastSquares while only having the labels ℓ_i and the points $\vec{x}_i$.

When the function $\vec{f}(\vec{x}, \vec{y}) : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^m$ takes in vectors and outputs a vector, the relevant derivatives for linearization are also represented by matrices:

$$D_{\vec{x}}\vec{f} = \begin{bmatrix} \frac{\partial f_1}{\partial x[1]} & \cdots & \frac{\partial f_1}{\partial x[n]} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x[1]} & \cdots & \frac{\partial f_m}{\partial x[n]} \end{bmatrix}.$$

$$D_{\vec{y}}\vec{f} = \begin{bmatrix} \frac{\partial f_1}{\partial y[1]} & \cdots & \frac{\partial f_1}{\partial y[k]} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial y[1]} & \cdots & \frac{\partial f_m}{\partial y[k]} \end{bmatrix}.$$

where

$$\vec{x} = \begin{bmatrix} x[1] \\ \vdots \\ x[n] \end{bmatrix} \vec{y} = \begin{bmatrix} y[1] \\ \vdots \\ y[n] \end{bmatrix}.$$

Then, the linearization (first-order expansion) becomes

$$\vec{f}(\vec{x}, \vec{y}) \approx \vec{f}(\vec{x}_0, \vec{y}_0) + D_{\vec{x}}\vec{f} \cdot (\vec{x} - \vec{x}_0) + D_{\vec{y}}\vec{f} \cdot (\vec{y} - \vec{y}_0). \tag{2}$$

(a) Now, suppose we wanted to approximate the cost for each data point

$$c_i(\vec{w}) = c(\vec{x}_i^\top \vec{w}, \ell_i) \tag{3}$$

where

$$\vec{w} = \begin{bmatrix} w[1] \\ \vdots \\ w[n] \end{bmatrix}$$

in the neighborhood of a weight vector $\vec{w}_*$. Our goal is to write out the first-order expression for approximating the cost function $c_i(\vec{w}_* + \vec{\delta w})$. This should be something in vector/matrix form like you have seen for the approximation of nonlinear systems by linear systems. We don't want to take any second derivatives just yet — only first derivatives. We have outlined a skeleton for the derivation with some parts missing. Follow the guidelines in each sub-section.

i) Comparing to eq. (2), we know that $c_i(\vec{w}_* + \vec{\delta w}) \approx c_i(\vec{w}_*) + \frac{d}{d\vec{w}} c_i(\vec{w}_*)\vec{\delta w}$. **Write out the vector form of** $\frac{d}{d\vec{w}} c_i(\vec{w}_*)$.

Solution:

$$\frac{d}{d\vec{w}} c_i(\vec{w}_*) = \begin{bmatrix} \frac{\partial c_i(\vec{w}_*)}{\partial w[1]} & \cdots & \frac{\partial c_i(\vec{w}_*)}{\partial w[n]} \end{bmatrix}$$

ii) **Write out the partial derivatives of** $c_i(\vec{w})$ **with respect to** $w[g]$**, the** g^{th} **component of** $\vec{w}$**.** *(HINT: Use the linearity of derivatives and sums to compute the partial derivatives with respect to each of the* $w[g]$ *terms. Don't forget the chain rule and the fact that* $\vec{x}_i^\top \vec{w} = \sum_{j=1}^n x_i[j]w[j] = x_i[g]w[g] + \sum_{j \neq g} x_i[j]w[j]$*.)*

Solution:

Using the hint, we calculate the partial derivative with respect to each $w[g]$ term.
Using the chain rule,

$$\begin{aligned}\frac{d}{d\vec{w}}c_i(\vec{w})[g] &= \frac{\partial}{\partial w[g]}c_i(\vec{w}) \\ &= \frac{\partial}{\partial w[g]}c(\vec{x}_i^\top\vec{w},\ell_i) \\ &= \frac{d}{d(\vec{x}^\top\vec{w})}c(\vec{x}_i^\top\vec{w},\ell_i)\frac{\partial}{\partial w[g]}(\vec{x}_i^\top\vec{w}) \\ &= c'(\vec{x}_i^\top\vec{w},\ell_i)\frac{\partial}{\partial w[g]}\left(x_i[g]w[g]+\sum_{j\neq g}^{n}x_i[j]w[j]\right) \\ &= c'(\vec{x}_i^\top\vec{w},\ell_i)x_i[g]\end{aligned}$$

Note that $c'(\vec{x}_i^\top\vec{w},\ell_i) = \frac{d}{d(\vec{x_i}^\top\vec{w})}c(\vec{x}_i^\top\vec{w},\ell_i)$.

iii) With what you had above, **can you fill in the missing part to express the row vector $\frac{d}{d\vec{w}}c_i(\vec{w})$?**

$$\frac{d}{d\vec{w}}c_i(\vec{w}) = c'(\vec{x}_i^\top\vec{w},\ell_i)\_\_\_\_\_$$

Solution:

$$\frac{d}{d\vec{w}}c_i(\vec{w}) = c'(\vec{x}_i^\top\vec{w},\ell_i)\vec{x}_i^\top$$

(b) Now, we want a better approximation that includes second derivatives. For a general function, we would look for

$$f(\vec{x}_0+\vec{\delta x}) \approx f(\vec{x}_0)+f'(\vec{x}_0)\vec{\delta x}+\frac{1}{2}\vec{\delta x}^{\top}f''(\vec{x}_0)\vec{\delta x} \tag{4}$$

where $f'(\vec{x}_0)$ is an appropriate row vector and, as you've seen in the note, $f''(\vec{x}_0)$ is called Hessian that represents the second derivatives.

i) Comparing to eq. (4), we know that

$$c_i(\vec{w}_*+\vec{\delta w}) \approx c_i(\vec{w}_*)+\frac{d}{d\vec{w}}c_i(\vec{w}_*)\vec{\delta w}+\frac{1}{2}\vec{\delta w}^{\top}\frac{d^2}{d\vec{w}^2}c_i(\vec{w}_*)\vec{\delta w}$$

Write out the matrix form of $\frac{d^2}{d\vec{w}^2}c_i(\vec{w}_*)$.

Solution:

$$\frac{d^2}{d\vec{w}^2}c_i(\vec{w}_*) = \begin{bmatrix}\frac{\partial^2 c_i(\vec{w}_*)}{\partial w[1]\partial w[1]} & \cdots & \frac{\partial^2 c_i(\vec{w}_*)}{\partial w[1]\partial w[n]} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 c_i(\vec{w}_*)}{\partial w[n]\partial w[1]} & \cdots & \frac{\partial^2 c_i(\vec{w}_*)}{\partial w[n]\partial w[n]}\end{bmatrix}$$

ii) **Take the second derivatives of the cost $c_i(\vec{w})$, i.e. solve for $\frac{\partial^2 c_i(\vec{w})}{\partial w[g]\partial w[h]}$.**
(HINT: You should use the answer to part (a) and just take another derivative. Once again, use the linearity of derivatives and sums to compute the partial derivatives with respect to each of the $w[h]$ terms. This will give you $\frac{\partial^2}{\partial w[g]\partial w[h]}$. Don't forget the chain rule and again use the fact that $\vec{x}_i^\top\vec{w} = \sum_{j=1}^{n}x_i[j]w[j] = x_i[h]w[h]+\sum_{j\neq h}x_i[j]w[j]$.)

Solution: Proceeding in a similar manner as above, let us find $\frac{\partial^2 c_i(\vec{w})}{\partial w[g]\partial w[h]}$.

$$\begin{aligned}\frac{d^2}{d\vec{w}^2}c_i(\vec{w})[g,h] &= \frac{\partial^2}{\partial w[g]\partial w[h]}c_i(\vec{w}) = \frac{\partial}{\partial w[h]}\left(\frac{d}{d\vec{w}}c_i(\vec{w})[g]\right)\\ &= \frac{\partial}{\partial w[h]}c'(\vec{x_i}^\top\vec{w},\ell_i)x_i[g]\\ &= c''(\vec{x_i}^\top\vec{w},\ell_i)\frac{\partial}{\partial w[h]}(\vec{x_i}^\top\vec{w})x_i[g]\\ &= c''(\vec{x_i}^\top\vec{w},\ell_i)x_i[g]x_i[h]\end{aligned}$$

Note that $c''(\vec{x_i}^\top\vec{w},\ell_i) = \frac{d^2}{d(\vec{x_i}^\top\vec{w})^2}c(\vec{x_i}^\top\vec{w},\ell_i)$.

iii) The expression in part (ii) is for the $[g,h]$-th component of the second derivative. $\frac{1}{2}$ times this times $\delta\vec{w}[g]$ times $\delta\vec{w}[h]$ would give us that component's contribution to the second-derivative term in the approximation, and we have to sum this up over all g and h to get the total contribution of the second-derivative term in the approximation. Now, we want to group terms to restructure this into matrix-vector form by utilizing the outer-product form of matrix multiplication. **What should the space in the following expression be filled with?**

$$\frac{d^2}{d\vec{w}^2}c_i(\vec{w}) = c''(\vec{x_i}^\top\vec{w},\ell_i)\_\_\_\_\_$$

Solution:

$$\frac{d^2}{d\vec{w}^2}c_i(\vec{w}) = c''(\vec{x_i}^\top\vec{w},\ell_i)\vec{x_i}\vec{x_i}^\top$$

(c) Now we have successfully expressed the second order approximation of $c_i(\vec{w}_* + \delta\vec{w})$. Since we eventually want to minimize the total cost $c_{total}(\vec{w}) = \sum_{i=1}^m c_i(\vec{w})$, **can you write out the second order approximation of $c_{total}(\vec{w}_* + \delta\vec{w})$ using results from (a) and (b)?**

Solution: From previous parts, we get

$$\begin{aligned}c_i(\vec{w}_* + \delta\vec{w}) &\approx c_i(\vec{w}_*) + \frac{d}{d\vec{w}}c_i(\vec{w}_*)\delta\vec{w} + \frac{1}{2}\delta\vec{w}^\top\frac{d^2}{d\vec{w}^2}c_i(\vec{w}_*)\delta\vec{w}\\ &= c_i(\vec{w}_*) + c'(\vec{x_i}^\top\vec{w}_*,\ell_i)\vec{x_i}^\top\delta\vec{w} + \frac{1}{2}\delta\vec{w}^\top c''(\vec{x_i}^\top\vec{w}_*,\ell_i)\vec{x_i}\vec{x_i}^\top\delta\vec{w}\end{aligned}$$

Based on the linearity of derivatives, to get the second order approximation of $c_{total}(\vec{w}_* + \delta\vec{w})$, we just sum up the second order approximation for each $c_i(\vec{w}_* + \delta\vec{w})$:

$$c_{total}(\vec{w}_* + \delta\vec{w}) = \sum_{i=1}^m c_i(\vec{w}_* + \delta\vec{w}) \approx \sum_{i=1}^m \left[c_i(\vec{w}_*) + c'(\vec{x_i}^\top\vec{w}_*,\ell_i)\vec{x_i}^\top\delta\vec{w} + \frac{1}{2}\delta\vec{w}^\top c''(\vec{x_i}^\top\vec{w}_*,\ell_i)\vec{x_i}\vec{x_i}^\top\delta\vec{w}\right]$$

(d) Now in this part, we want to re-write $c_{total}(\vec{w}_* + \delta\vec{w})$ in form of $C + \sum_{i=1}^m(\vec{q_i}^\top\delta\vec{w} - b_i)^2$.

i) Let's first rewrite a general second order polynomial $f(x) = ax^2 + bx + c$ in the form of $f(x) = r + (px+q)^2$. **Find p, q, r in terms of a, b, c.** This procedure is called "completing the square". Then, **use this to argue that**

$$\arg\min_x ax^2 + bx + c = \arg\min_x (px+q)^2$$

Solution: $ax^2+bx+c = c - \frac{b^2}{4a} + (\sqrt{a}x + \frac{b}{2\sqrt{a}})^2$. Therefore $p = \sqrt{a}$, $q = \frac{b}{2\sqrt{a}}$, $r = c - \frac{b^2}{4a}$. Since r is a constant, $\arg\min_x r + (px+q)^2 = \arg\min_x (px+q)^2$, hence we have

$$\arg\min_x ax^2 + bx + c = \arg\min_x (px+q)^2$$

ii) Now rewrite $c_{total}(\vec{w}_* + \vec{\delta w})$ in the form of $C + \sum_{i=1}^m (\vec{q}_i^\top \vec{\delta w} - b_i)^2$. **What are C, q_i, and b_i?**

Solution:

$$\begin{aligned}
c_{total}(\vec{w}_* + \vec{\delta w}) &\approx \sum_{i=1}^m \left[c_i(\vec{w}_*) + c'(\vec{x}_i^\top \vec{w}_*, \ell_i)\vec{x}_i^\top \vec{\delta w} + \frac{1}{2}\vec{\delta w}^\top c''(\vec{x}_i^\top \vec{w}_*, \ell_i)\vec{x}_i\vec{x}_i^\top \vec{\delta w} \right] \\
&= \sum_{i=1}^m \left[c_i(\vec{w}_*) - \frac{\left(c'(\vec{x}_i^\top \vec{w}_*, \ell_i)\right)^2}{4\frac{1}{2}c''(\vec{x}_i^\top \vec{w}_*, \ell_i)} + \left(\sqrt{\frac{1}{2}c''(\vec{x}_i^\top \vec{w}_*, \ell_i)}\vec{x}_i^\top \vec{\delta w} - \frac{-c'(\vec{x}_i^\top \vec{w}_*, \ell_i)}{\sqrt{2c''(\vec{x}_i^\top \vec{w}_*, \ell_i)}} \right)^2 \right] \\
&= \sum_{i=1}^m \left[c_i(\vec{w}_*) - \frac{\left(c'(\vec{x}_i^\top \vec{w}_*, \ell_i)\right)^2}{2c''(\vec{x}_i^\top \vec{w}_*, \ell_i)} \right] + \sum_{i=1}^m \left[\left(\sqrt{\frac{1}{2}c''(\vec{x}_i^\top \vec{w}_*, \ell_i)}\vec{x}_i^\top \vec{\delta w} - \frac{-c'(\vec{x}_i^\top \vec{w}_*, \ell_i)}{\sqrt{2c''(\vec{x}_i^\top \vec{w}_*, \ell_i)}} \right)^2 \right]
\end{aligned}$$

Comparing to the expected format we can see that

$$C = \sum_{i=1}^m \left[c_i(\vec{w}_*) - \frac{\left(c'(\vec{x}_i^\top \vec{w}_*, \ell_i)\right)^2}{2c''(\vec{x}_i^\top \vec{w}_*, \ell_i)} \right] \qquad \vec{q}_i = \sqrt{\frac{1}{2}c''(\vec{x}_i^\top \vec{w}_*, \ell_i)}\vec{x}_i \qquad b_i = \frac{-c'(\vec{x}_i^\top \vec{w}_*, \ell_i)}{\sqrt{2c''(\vec{x}_i^\top \vec{w}_*, \ell_i)}}$$

(e) Consider a least squares problem:

$$\vec{x^\star} = \arg\min_{\vec{x}} \left\| A\vec{x} - \vec{b} \right\|^2$$

Show that:

$$\left\| A\vec{x} - \vec{b} \right\|^2 = \sum_{i=1}^m (\vec{a}_i^\top \vec{x} - b_i)^2$$

where

$$A = \begin{bmatrix} - & \vec{a}_1^\top & - \\ - & \vec{a}_2^\top & - \\ & \vdots & \\ - & \vec{a}_m^\top & - \end{bmatrix}.$$

Use this to intepret your expression from Part (d) as a standard least squares problem. What are the rows of A?

Solution: $\left\| A\vec{x} - \vec{b} \right\|^2$ is by definition equal to the sum of all entries squared of the vector $A\vec{x} - \vec{b}$. Therefore $\left\| A\vec{x} - \vec{b} \right\|^2 = \sum_{i=1}^m (\vec{a}_i^\top \vec{x} - b_i)^2$. Matching terms with our expression of $c_{total}(\vec{w} + \vec{\delta w})$ in

Part (d), we get

$$A = \begin{pmatrix} \sqrt{\frac{1}{2}c''(\vec{x_1}^\top \vec{w}_*, \ell_1)}\vec{x_1}^\top \\ \sqrt{\frac{1}{2}c''(\vec{x_2}^\top \vec{w}_*, \ell_2)}\vec{x_2}^\top \\ \vdots \\ \sqrt{\frac{1}{2}c''(\vec{x_m}^\top \vec{w}_*, \ell_m)}\vec{x_m}^\top \end{pmatrix}$$

.

(f) Consider the following cost functions:
squared error: $c^+_{sq}(p) = (p-1)^2,\ c^-_{sq}(p) = (p+1)^2$;
exponential: $c^+_{exp}(p) = e^{-p},\ c^-_{exp}(p) = e^p$;
and logistic: $c^+_{logistic}(p) = \ln\left(1+e^{-p}\right),\ c^-_{logistic}(p) = \ln(1+e^p)$.
Compute the first and second derivatives of the above expressions with respect to p.

Solution: First Derivatives:

$$\frac{d}{dp}c^+_{sq}(p) = 2(p-1)$$
$$\frac{d}{dp}c^-_{sq}(p) = 2(p+1)$$
$$\frac{d}{dp}c^+_{exp}(p) = -e^{-p}$$
$$\frac{d}{dp}c^-_{exp}(p) = e^p$$
$$\frac{d}{dp}c^+_{logistic}(p) = -\frac{e^{-p}}{1+e^{-p}}$$
$$\frac{d}{dp}c^-_{logistic}(p) = \frac{e^p}{1+e^p}$$

Notice that these are pretty cheap to compute, given that we have to compute the original loss functions in the first place.
Second Derivatives:

$$\frac{d^2}{dp^2}c^+_{sq}(p) = 2$$
$$\frac{d^2}{dp^2}c^-_{sq}(p) = 2$$
$$\frac{d^2}{dp^2}c^+_{exp}(p) = e^{-p}$$
$$\frac{d^2}{dp^2}c^-_{exp}(p) = e^p$$
$$\frac{d^2}{dp^2}c^+_{logistic}(p) = \frac{e^{-p}}{(1+e^{-p})^2}$$
$$\frac{d^2}{dp^2}c^-_{logistic}(p) = \frac{e^p}{(1+e^p)^2}$$

Notice that all of these second derivatives are positive. Moreover, calculating them takes essentially no more work than getting the first derivatives. In particular, it is useful to note that

$$\frac{d^2}{dp^2}c^+_{logistic}(p) = \frac{e^{-p}}{(1+e^{-p})^2} = |\frac{d}{dp}c^+_{logistic}(p)|(1-|\frac{d}{dp}c^+_{logistic}(p)|)$$
$$\frac{d^2}{dp^2}c^-_{logistic}(p) = \frac{e^{p}}{(1+e^{p})^2} = |\frac{d}{dp}c^-_{logistic}(p)|(1-|\frac{d}{dp}c^-_{logistic}(p)|)$$

so the basic nature of the logistic loss' second derivative becomes more clear. If the first derivative has magnitude $\frac{1}{2}$, this second derivative is maximized — that happens when the prediction p is 0 or maximally uncertain. The second derivative for logistic loss shrinks away from there.

When you take 70, this particular form $p(1-p)$ will start to ring a bell. That sound is the gateway to understanding a different reason for why logistic regression is popular in practice — but for that, you need to understand Probability. It is a glorious coincidence that something so natural from an optimization point of view also turns out to have useful interpretations involving the language of probability. You will understand this after 126.

(g) Run the Jupyter Notebook and answer the following questions.

i) **In Example 2, why does mean classification fail?**

Solution: The mean classifier misclassifies some points because there is some information in the distribution that can't be accurately captured by the mean of the data points of each category.

ii) **In Example 3, for what data distributions does ordinary least squares fail?**

Solution: When there are extreme outliers in the dataset, the decision boundary in this case will be "pulled" away from the desired location, thus least squares would fail.

iii) Run the code cells in Example 4. By performing updates to $\vec{w}$ according to what you derived in previous parts of the question, **how many iterations does it take for exponential and logistic regression to converge?**

Solution: You should see that the decision boundaries are almost fixed after 3-4 iterations. These iterated least-squares approaches are very fast to converge in general. This is why in practice, logistic or exponential regression costs almost the same to run as ordinary least squares.

Congratulations! You now know the basic optimization-theoretic perspective on logistic regression. After you understand the probability material in 70 and 126, you can understand the probabilistic perspective on it as well. After you understand the optimization material in 127, you will understand even more about the optimization-theoretic perspective on the problem including why this approach actually converges.

3. Extending Orthonormality to Complex Vectors

So far in the course, we have only dealt with real vectors. However, it is often useful to also think about complex vectors, as you'll soon see with the DFT. In this problem, we will extend several important properties of orthonormal matrices to the complex case.

The main difference is that the normal Euclidean inner product is no longer a valid metric, and we must define a new complex inner product as

$$\langle \vec{u}, \vec{v} \rangle = \overline{\vec{v}^T} \vec{u} = \vec{v}^* \vec{u} = \sum_{i=1}^{k} u_i \overline{v_i},$$

where the * operation will complex conjugate and transpose its argument (order doesn't matter), and is aptly called the *conjugate transpose*. Note that for real numbers, the complex inner product simplifies to the real inner product. In all the theorems you've seen in this class, you can replace every inner product with the complex inner product to show an analogous result for complex vectors: least squares becomes $\hat{x} = (A^*A)^{-1}A^*\vec{b}$, upper triangularization becomes $A = UTU^*$, Spectral Theorem becomes $A = U\Lambda U^*$, SVD becomes $A = U\Sigma V^*$.

(a) To get some practice computing complex inner products, what is $\left\langle \begin{bmatrix} 1+j \\ 2 \end{bmatrix}, \begin{bmatrix} -3-j \\ 2+j \end{bmatrix} \right\rangle$ and $\left\langle \begin{bmatrix} -3-j \\ 2+j \end{bmatrix}, \begin{bmatrix} 1+j \\ 2 \end{bmatrix} \right\rangle$? Does the order of the vectors in the complex inner product matter i.e. is it commutative?

Solution:

$$\begin{aligned}
\left\langle \begin{bmatrix} 1+j \\ 2 \end{bmatrix}, \begin{bmatrix} -3-j \\ 2+j \end{bmatrix} \right\rangle &= (1+j)\overline{(-3-j)} + (2)\overline{(2+j)} \\
&= (1+j)(-3+j) + 2(2-j) \\
&= -3+j-3j-1+4-2j = -4j \\
\left\langle \begin{bmatrix} -3-j \\ 2+j \end{bmatrix}, \begin{bmatrix} 1+j \\ 2 \end{bmatrix} \right\rangle &= (-3-j)\overline{(1+j)} + (2+j)\overline{(2)} \\
&= (-3-j)(1-j) + (2+j)(2) \\
&= -3+3j-j-1+4+2j = 4j
\end{aligned}$$

The two inner products are different so clearly the complex inner product is not commutative and the order of the vectors matters. In fact, when you swap the arguments for the inner product, you will get the complex conjugate result, which you see an example of above.

(b) Let $U = \begin{bmatrix} \vec{u}_1 & \cdots & \vec{u}_n \end{bmatrix}$ be an n by n complex matrix, where its columns $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n$ form an orthonormal basis for $\mathbb{C}^n$, i.e.

$$\vec{u}_i^* \vec{u}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Such a complex matrix is called *unitary* in math literature, to distinguish from real orthonormal matrices. **Show that $U^{-1} = U^*$, where U^* is the conjugate transpose of U.**

Solution: By definition, $U^{-1}U = I$. We want to show that U^* satisfies it so let's write down U^* first:

$$U^* = \begin{bmatrix} \vec{u}_1^* \\ \vdots \\ \vec{u}_n^* \end{bmatrix}, \tag{5}$$

where $\vec{u}_1, \ldots, \vec{u}_n$ are the column vectors of U. Then, the entry at the i-th row and j-th column of U^*U should be $\vec{u}_i^* \vec{u}_j$. If we write down the general form for each element of U^*U:

$$(U^*U)_{ij} = \vec{u}_i^* \vec{u}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases},$$

which is the identity matrix, since $\vec{u}_1, \ldots, \vec{u}_n$ is an orthonormal basis.

Now for any square matrices A, B such that $AB = I$, right multiplying by B^{-1} gives $ABB^{-1} = IB^{-1}$ so $A = B^{-1}$. B^{-1} must exist since $\det(A)\det(B) = \det(I) \neq 0$ so $\det(B) \neq 0$. Thus since we showed $U^*U = I$, then $U^* = U^{-1}$.

(c) **Show that U preserves complex inner products, i.e. if $\vec{v}, \vec{w}$ are vectors of length n, then**

$$\langle \vec{v}, \vec{w} \rangle = \langle U\vec{v}, U\vec{w} \rangle .$$

HINT: Note that $(AB)^* = B^*A^*$. This is since $(AB)^* = \overline{(AB)^T} = \overline{B^T A^T} = \overline{B^T}\ \overline{A^T} = B^*A^*$

Solution: For this question, we want to show that:

$$\langle \vec{v}, \vec{w} \rangle = \vec{w}^* \vec{v} = \langle U\vec{v}, U\vec{w} \rangle$$

Using the definition of complex inner products we can write:

$$\langle U\vec{v}, U\vec{w} \rangle = (U\vec{w})^* U\vec{v}$$

Using the form for the complex conjugate of a matrix-vector product as stated in the problem:

$$(U\vec{w})^* U\vec{v} = \vec{w}^* U^* U\vec{v}$$

From the previous problem we know that $U^*U = I$. Therefore:

$$\langle U\vec{v}, U\vec{w} \rangle = \vec{w}^* \vec{v} = \langle \vec{v}, \vec{w} \rangle.$$

(d) **Show that if $\vec{u}_1, \ldots, \vec{u}_n$ are the columns of a unitary matrix U, they must be linearly independent.**

(Hint: Suppose $\vec{w} = \sum_{i=1}^n \alpha_i \vec{u}_i$, then first show that $\alpha_i = \langle \vec{w}, \vec{u}_i \rangle$. From here ask yourself whether a nonzero linear combination of the $\{\vec{u}_i\}$ could ever be identically zero.)

This basic fact shows how orthogonality is a very nice special case of linear independence.

Solution: Suppose they are not linearly independent, then there exist $\alpha_1, \ldots, \alpha_n \in \mathbb{C}$ such that $\vec{w} = \sum_{i=1}^n \alpha_i \vec{u}_i = \vec{0}$, while at least one of α_i is non-zero. We can then take the inner product of both sides with $\vec{u}_j$, for all j:

$$\langle \vec{w}, \vec{u}_j \rangle = \left\langle \sum_{i=1}^n \alpha_i \vec{u}_i, \vec{u}_j \right\rangle = \sum_{i=1}^n \alpha_i \langle \vec{u}_i, \vec{u}_j \rangle = \alpha_j .$$

Since $\vec{u}_1, \ldots, \vec{u}_n$ form an orthonormal basis, we know that $\langle \vec{u}_i, \vec{u}_j \rangle$ will be 1 when $i = j$ and 0 otherwise, which is why only α_j survives in the above summation. Since $\vec{w} = \vec{0}$, then α_j should be 0 for all inner products $\langle \vec{u}_j, \vec{w} \rangle$. However, this is a contradiction to our assumption that at least one of the α_i is non-zero. Therefore, $\vec{u}_1, \ldots, \vec{u}_n$ are linearly independent.

This confirms what we know — that orthonormality is a particularly robust guarantee of linear independence.

(e) Now let V be another $n \times n$ matrix, where the columns of the matrix form an orthonormal basis for $\mathbb{C}^n$, i.e. V is unitary. **Show that the columns of the product UV also form an orthonormal basis for $\mathbb{C}^n$.**)

Solution: Since V is a unitary matrix, we have $V^*V = I$. To show that the columns of UV also form an orthonormal basis, we could write down its conjugate transpose, $(UV)^*$, and apply it to UV:

$$(UV)^*(UV) = V^*U^*UV = V^*V = I\,,$$

which means the columns of UV form an orthonormal basis for $\mathbb{C}^n$.

(f) We can also extend the idea of symmetric matrices to complex vectors, though we again will need to replace the transpose with the conjugate transpose. If $M = M^*$, then M is called a *Hermitian* matrix, and the Spectral Theorem will say it can be diagonalized by a unitary U, i.e. $M = U\Lambda U^*$, where Λ is a diagonal matrix with the eigenvalues along the diagonal. **Show that M has real eigenvalues.**

HINT: Use the fact that $M^* = M$.

Solution: We first calculate M^*

$$M^* = (U\Lambda U^*)^* = U\Lambda^* U^*.$$

Since $M^* = M = U\Lambda U^*$, this means that $\Lambda^* = \Lambda$. The only case where this is true is when all the elements of Λ are real, which means the eigenvalues of M are always real.

4. Roots of Unity

An Nth root of unity is a complex number ω satisfying the equation $\omega^N = 1$ (or equivalently $\omega^N - 1 = 0$). In this problem we explore some properties of the roots of unity, as they end up being essential for the DFT.

(a) **Show that the polynomial $z^N - 1$ factors as**

$$z^N - 1 = (z-1)\left(\sum_{k=0}^{N-1} z^k\right).$$

Solution:

$$(z-1)\left(\sum_{k=0}^{N-1} z^k\right) = \sum_{k=1}^{N} z^k - \sum_{k=0}^{N-1} z^k = z^N - z^0 = z^N - 1$$

(b) **Show that any complex number of the form $\omega_N^k = e^{j\frac{2\pi}{N}k}$ for $k \in \mathbb{Z}$ is an N-th root of unity.** From here on, let $\omega_N = e^{j\frac{2\pi}{N}}$.

Solution:

$$\omega_N^{kN} = \left(e^{j\frac{2\pi}{N}k}\right)^N = e^{j2\pi k} = e^0 = 1$$

This means that the N numbers $\omega_N^0, \omega_N^1, \omega_N^2, \ldots, \omega_N^{N-1}$ are the solutions to the equation $z^N = 1$ and hence roots of the polynomial $z^N - 1$.

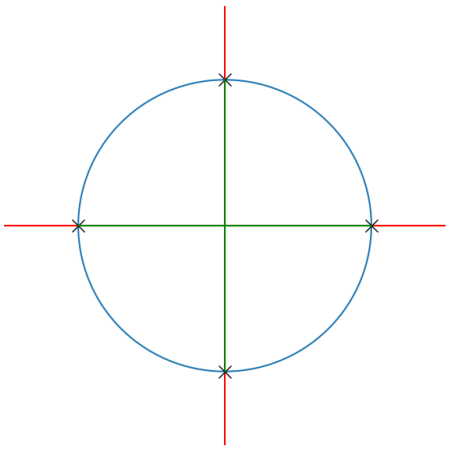
(c) For a given integer $N \geq 2$, using the previous parts, **give the complex roots of the polynomial $1 + z + z^2 + ... + z^{N-1}$.**

Solution: $(z-1)(1+z+z^2+...+z^{N-1}) = z^N - 1$, and we just showed that the roots of $z^N - 1$ are the $\omega^k = e^{j2\pi k}$ for $k = 0, \ldots, N-1$, therefore the roots of $z \mapsto 1 + z + z^2 + ... + z^{N-1}$ are the $\omega_N^k = e^{j\frac{2\pi k}{N}}$ for $k = 1, \ldots, N-1$.

We can see this by factoring and matching factors together. A polynomial with a leading coefficient of 1 can be factored into its roots. So $z^N - 1 = (z - \omega_N^0)(z - \omega_N^1)(z - \omega_N^2)\cdots(z - \omega_N^{N-1}) = \prod_{k=0}^{N-1}(z - \omega_N^k) = (z-1)\prod_{k=1}^{N-1}(z - \omega_N^k)$. Dividing both sides by $z - 1$ gives us what we want.

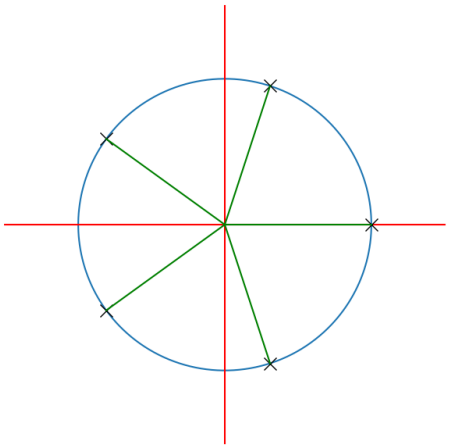
(d) **What are the fourth roots of unity? Draw the fourth roots of unity in the complex plane. Where do they lie in relation to the unit circle?**

Solution: Using the formula for the roots of unity from part (b), $\omega_4^0 = e^0 = 1, \omega_4^1 = e^{j\frac{\pi}{2}} = j, \omega_4^2 = e^{j\frac{2\pi}{2}} = -1, \omega_4^3 = e^{j\frac{3\pi}{2}} = -j$. All roots of unity must be on the unit circle since they have magnitude 1. From the definition of the roots of unity, we know that each root of unity is $2\pi/N = \pi/2$ radians apart, and this can be seen graphicaly in the plot below.



(e) **What are the fifth roots of unity? Draw the fifth roots of unity in the complex plane.**

Solution: Using the formula for the roots of unity from part (b), $\omega_5^0 = e^0 = 1, \omega_5^1 = e^{j\frac{2\pi}{5}}, \omega_5^2 = e^{j\frac{4\pi}{5}}, \omega_5^3 = e^{j\frac{6\pi}{5}}, \omega_5^4 = e^{j\frac{8\pi}{5}}$. Again, we know that each root of unity should be $2\pi/5$ radians or $72°$ apart by the definition, and it can be seen graphically below.



(f) **For $N = 5$, $\omega_5 = e^{j\frac{2\pi}{5}}$, simplify ω_5^{42} such that the exponent is less than 5 and greater than 0.**

Solution:

$$\omega_5^{42} = \omega_5^{8\cdot 5}\omega_5^2 = (\omega_5^5)^8\omega_5^2 = 1^8\omega_5^2 = \omega_5^2$$

Every 5 powers of the 5th root of unity will be 1, so it simplifies to the remainder of 42 divided by 5.

(g) Let's generalize what you saw in the previous part. **Prove that $\omega_N^{k+N} = \omega_N^k$ for all integers k, both positive and negative.** This shows that the roots of unity have a periodic structure.

Solution:

$$\omega_N^{k+N} = \omega_N^k \omega_N^N = \omega_N^k \cdot 1 = \omega_N^k$$

(h) **What is the complex conjugate of ω_5 in terms of the 5th roots of unity? What is the complex conjugate of ω_5^{42} in terms of the 5th roots of unity? What is the complex conjugate of ω_5^4 in terms of the 5th roots of unity?**

Solution: Using part (g),

$$\bar{\omega}_5 = \omega_5^{-1} = \omega_5^4$$
$$\bar{\omega}_5^{42} = \omega_5^{-42} = \omega_5^3$$
$$\bar{\omega}_5^4 = \omega_5^{-4} = \omega_5$$

Notice here that we can think about going around the circle of roots of unity, since they are periodic and wrap around to where they started.

This is something called modulo arithmetic and is naturally connected to cycles like this. It is traditionally viewed as taking the remainder (in this case after dividing by $N = 5$), however some people get confused when asked to take the remainder of a negative number divided by a positive one. Here, we just remember that we can turn a negative number between $-(N-1)$ and -1 into a positive number by adding N to it.

You will learn a lot more about modulo arithmetic in CS 70 — here in 16B, you just get a teaser because it emerges naturally when thinking about the roots of unity and their powers.

(i) **Compute $\sum_{m=0}^{N-1} \omega_N^{km}$ where ω_N is an Nth root of unity.** Does the answer make sense in terms of the plot you drew?

Solution: If $\omega_N^k = 1$, then this is easy: we have

$$\sum_{m=0}^{N-1} \omega_N^0 = \sum_{m=0}^{N-1} 1 = N.$$

This happens whenever k is a multiple of N. For all other k, we know $\omega_N^k \neq 1$. Consequently, we can use the formula we found in part (a) to write

$$\sum_{m=0}^{N-1} \omega_N^{km} = \frac{\omega_N^{kN} - 1}{\omega_N - 1} = 0$$

since ω_N is a root of unity and so $\omega_N^N = 1$ and so $\omega_N^{kN} = 1$ as well. This makes intuitive sense because all the roots of unity are spaced evenly around the circle. Therefore summing them up, we have to get zero by symmetry.

5. Discrete Fourier Transform (DFT)

In order to get practice with calculating the Discrete Fourier Transform (DFT), this problem will have you calculate the DFT for a few variations on a cosine signal.

Consider a sampled signal that is a function of discrete time $x[t]$. We can represent it as a vector of discrete samples over time $\vec{x}$, of length N.

$$\vec{x} = \begin{bmatrix} x[0] & \dots & x[N-1] \end{bmatrix}^T \tag{6}$$

We can represent the DFT basis with the matrix U, and it is given by

$$U = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{j\frac{2\pi}{N}} & e^{j\frac{2\pi(2)}{N}} & \cdots & e^{j\frac{2\pi(N-1)}{N}} \\ 1 & e^{j\frac{2\pi(2)}{N}} & e^{j\frac{2\pi(4)}{N}} & \cdots & e^{j\frac{2\pi 2(N-1)}{N}} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{j\frac{2\pi(N-1)}{N}} & e^{j\frac{2\pi 2(N-1)}{N}} & \cdots & e^{j\frac{2\pi(N-1)(N-1)}{N}} \end{bmatrix}.$$

In other words, for the ij^{th} entry of U, $U_{ij} = \frac{1}{\sqrt{N}} e^{j\frac{2\pi ij}{N}}$. From this, we can see that the DFT basis matrix is symmetric, so $U = U^T$. Another very important property of the DFT basis is that U is orthonormal, so $U^*U = I$. We want to find the coordinates of $\vec{x}$ in the DFT basis, and we know these coordinates are given by

$$\vec{X} = U^{-1}\vec{x}.$$

We call the components of $\vec{X}$ the *DFT coefficients* of the time-domain signal $\vec{x}$. We can think of the components of $\vec{X}$ as weights that represent $\vec{x}$ in the DFT basis. As we will explore in the problem, each coefficient can be thought of as a measurement for which frequency is present in the signal.

You can use Numpy or other calculation tools to evaluate cosines or do matrix multiplication, but you will not get credit if you directly calculate the DFT using a function in Numpy. You must show your work to get credit.

(a) **What is U^{-1}?**

Solution: Since U is orthonormal, $U^{-1} = U^*$ which means we transpose and complex conjugate U. However, note that U is symmetric, so we just need to complex conjugate the matrix, meaning every exponent of the exponential becomes negative. Thus,

$$U^{-1} = U^* = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & e^{-j\frac{2\pi(2)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)}{N}} \\ 1 & e^{-j\frac{2\pi(2)}{N}} & e^{-j\frac{2\pi(2)(2)}{N}} & \cdots & e^{-j\frac{2\pi 2(N-1)}{N}} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{-j\frac{2\pi(N-1)}{N}} & e^{-j\frac{2\pi 2(N-1)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)(N-1)}{N}} \end{bmatrix}$$

(b) Let the columns of $U = \begin{bmatrix} \vec{u}_0 & \vec{u}_1 & \dots & \vec{u}_{N-1} \end{bmatrix}$. **Prove that $\overline{\vec{u}_k} = \vec{u}_{N-k}$ for $k = 1, 2, \dots, N-1$.**

Solution:

$$\overline{\vec{u}_k}[m] = \overline{e^{j\frac{2\pi mk}{N}}} = e^{-j\frac{2\pi mk}{N}} \tag{7}$$

Using the fact that any multiple of 2π won't affect the exponent,

$$e^{-j\frac{2\pi mk}{N}} = e^{j(2\pi m - \frac{2\pi mk}{N})} \tag{8}$$

$$= e^{j\frac{2\pi m(N-k)}{N}} = \vec{u}_{N-k}[m] \tag{9}$$

Since this holds for all m, then $\overline{\vec{u}_k} = \vec{u}_{N-k}$ when $k = 1, \ldots, N-1$. It doesn't hold for other k since then the indices of k and $N-k$ for the columns of U wouldn't be valid.

(c) **Decompose $\cos\left(\frac{2\pi}{7}n\right)$ into a sum of complex exponentials.**

Solution: From the inverse Euler formula,

$$\cos\left(\frac{2\pi}{7}n\right) = \frac{1}{2}e^{j\frac{2\pi n}{7}} + \frac{1}{2}e^{-j\frac{2\pi n}{7}}$$

(d) If $x_1[n] = \cos\left(\frac{2\pi n}{7}\right)$, **write $x_1[n]$ as a linear combination of $y_+[n] = e^{j\frac{2\pi n}{7}}$ and $y_-[n] = e^{-j\frac{2\pi n}{7}}$.**

Solution: Note that we can replace both complex exponentials with the given y functions, so

$$x_1[n] = \frac{1}{2}y_+[n] + \frac{1}{2}y_-[n]$$

(e) Now think of $\vec{x}_1$ as a length $N = 7$ vector which is $x_1[n]$ sampled at $n = 0, 1, \ldots, 6$. We do the same to similarly get $\vec{y}_+, \vec{y}_-$. **Write $\vec{y}_+$ and $\vec{y}_-$ in terms of the columns of $U = \begin{bmatrix}\vec{u}_0 & \vec{u}_1 & \ldots & \vec{u}_6\end{bmatrix}$.**
HINT: Use part (b).

Solution: We can notice that $\vec{u}_1$ is exactly $\frac{1}{\sqrt{N}}e^{j\frac{2\pi n}{7}}$ evaluated at the 7 sample points. This is exactly $\vec{y}_+$ up to scale, so $\vec{y}_+ = \sqrt{N}\vec{u}_1 = \sqrt{7}\vec{u}_1$. From part (b), we know that $\vec{u}_6 = \overline{\vec{u}_1}$ which is $\frac{1}{\sqrt{N}}e^{-j\frac{2\pi n}{7}}$ evaluated at the sample points. This is exactly a scaling of $\vec{y}_-$, so $\vec{y}_- = \sqrt{N}\vec{u}_6 = \sqrt{7}\vec{u}_6$.

(f) Using the last 3 parts, **compute the DFT coefficients $\vec{X}_1$ for signal $\vec{x}_1$.**

Solution: Combining the last 3 parts, we know $\vec{x}_1 = \frac{\sqrt{7}}{2}\vec{u}_1 + \frac{\sqrt{7}}{2}\vec{u}_6$. We also know that since the columns of U are $\vec{u}_0, \ldots, \vec{u}_6$, then

$$U^* = \frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & e^{-j\frac{2\pi(2)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)}{N}} \\ 1 & e^{-j\frac{2\pi(2)}{N}} & e^{-j\frac{2\pi(2)(2)}{N}} & \cdots & e^{-j\frac{2\pi 2(N-1)}{N}} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{-j\frac{2\pi(N-1)}{N}} & e^{-j\frac{2\pi 2(N-1)}{N}} & \cdots & e^{-j\frac{2\pi(N-1)(N-1)}{N}} \end{bmatrix} = \begin{bmatrix} \vec{u}_0^* \\ \vec{u}_1^* \\ \vec{u}_2^* \\ \vdots \\ \vec{u}_{N-1}^* \end{bmatrix} \tag{10}$$

Then to find our DFT coefficients,

$$\vec{X}_1 = U^* \vec{x}_1 \tag{11}$$

$$= \begin{bmatrix} \vec{u}_0^* \\ \vec{u}_1^* \\ \vdots \\ \vec{u}_6^* \end{bmatrix} \left(\frac{\sqrt{7}}{2}\vec{u}_1 + \frac{\sqrt{7}}{2}\vec{u}_6\right) \tag{12}$$

$$= \frac{\sqrt{7}}{2}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \frac{\sqrt{7}}{2}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\sqrt{7}}{2} \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{\sqrt{7}}{2} \end{bmatrix} \tag{13}$$

In the last equation, we use the orthonormality property of the DFT basis vectors, i.e. that

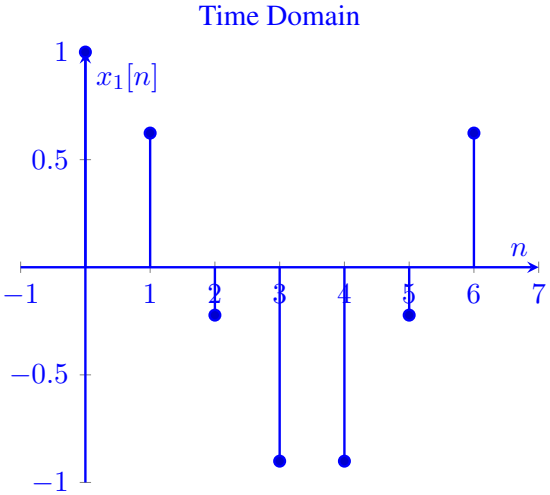
$$\vec{u}_i^* u_j = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases} \tag{14}$$

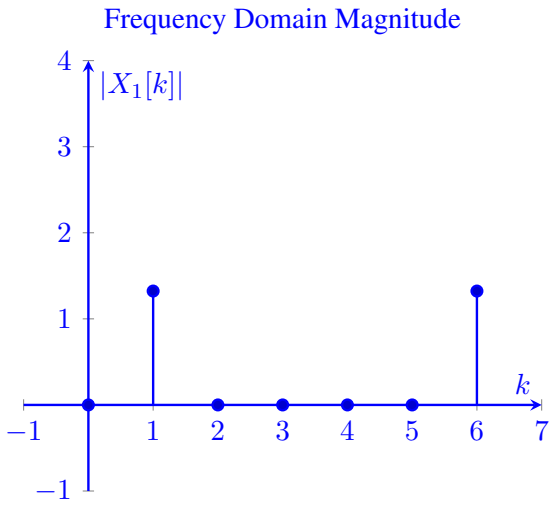
Equivalently, we can write the result as

$$X_1[k] = \begin{cases} \frac{\sqrt{7}}{2} & k = 1, 6 \\ 0 & k \neq 1, 6. \end{cases} \tag{15}$$

(g) **Plot the time domain representation of $x_1[n]$. Plot the magnitude, $|X_1[n]|$, and plot the phase, $\angle X_1[n]$, for the DFT representation $\vec{X}_1$.** You should notice that the DFT coefficients correspond to which frequencies were present in the original signal, which is why the DFT coefficients are called the *frequency domain* representation.

Solution:





Because the coefficients $\vec{X}_1$ in this case are all real-valued, the phase is zero-valued.

(h) We define $x_2[n] = \cos\left(\frac{4\pi}{7}n\right)$ for $N = 7$ samples $n \in \{0, 1, \ldots, 6\}$. **Compute the DFT coefficients $\vec{X}_2$ for signal $\vec{x}_2$.**

Solution: Following from the above, we can write $\vec{x}_2$, the vector of samples of the function, in terms of the columns of our DFT basis matrix U:

$$x_2[n] = \cos\left(\frac{2\pi(2)}{7}n\right) = \frac{1}{2}e^{j\frac{2\pi(2)}{7}n} + \frac{1}{2}e^{-j\frac{2\pi(2)}{7}n} \tag{16}$$

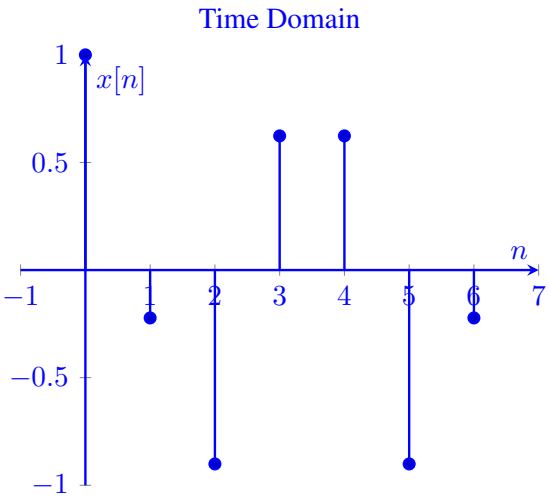
$$\vec{x}_2 = \frac{1}{2}(\sqrt{N}\vec{u}_2 + \sqrt{N}\,\overline{\vec{u}_2}) = \frac{\sqrt{7}}{2}(\vec{u}_2 + \vec{u}_5) \tag{17}$$

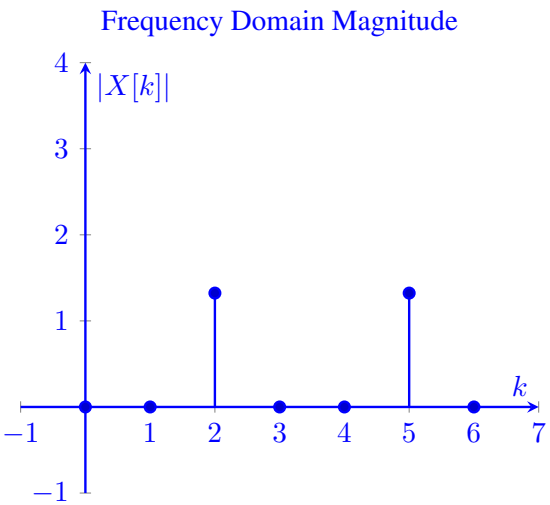
From this, we can see that our DFT coefficients vector $\vec{X}_2$ will only have non-zero values for rows $k = 2$ and $k = 5$. The elements of $\vec{X}_2$ can be written as

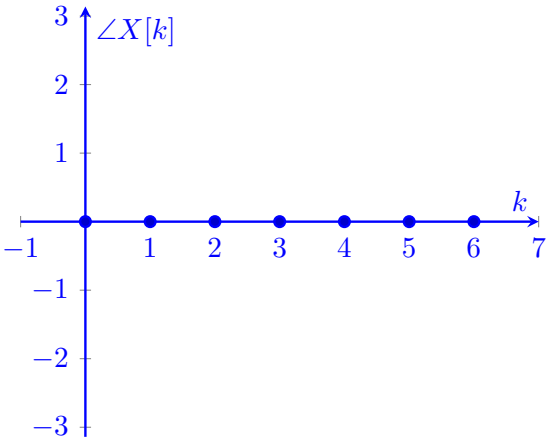
$$X_2[k] = \begin{cases} \frac{\sqrt{7}}{2} & k = 2, 5 \\ 0 & k \neq 2, 5. \end{cases} \tag{18}$$

(i) **Plot the time domain representation of $x_2[n]$. Plot the magnitude, $|X_2[n]|$, and plot the phase, $\angle X_2[n]$, for the DFT representation $\vec{X}_2$.**

Solution:







(j) To generalize this result, say we have some $p \in \{1, 2, 3\}$ which scales the frequency of our signal $\vec{x}_p$,

which we define as $x_p[n] = \cos\left(\frac{2\pi}{7}pn\right)$ for $N = 7$ samples $n \in \{0, 1, \ldots, 6\}$. **Compute the DFT coefficients $\vec{X}_p$ for signal $\vec{x}_p$ in terms of this scalar p.**

Solution: As in the last two problems, we can represent $\vec{x}_p$ in terms of the columns of the DFT basis matrix:

$$x_p[n] = \cos\left(\frac{2\pi(p)}{7}n\right) = \frac{1}{2}e^{j\frac{2\pi(p)}{7}n} + \frac{1}{2}e^{-j\frac{2\pi(p)}{7}n} \tag{19}$$

$$\vec{x}_p = \frac{1}{2}(\sqrt{7}\vec{u_p} + \sqrt{7}\,\overline{\vec{u}_p}) = \frac{\sqrt{7}}{2}(\vec{u_p} + \vec{u}_{7-p}) \tag{20}$$

The only nonzero entries in $\vec{X}_p$ will occur at $k = p$ and k = $7 - p$:

$$X_p[k] = \begin{cases} \frac{\sqrt{7}}{2} & k = p, 7-p \\ 0 & k \neq p, 7-p. \end{cases} \tag{21}$$

(k) Let's see what happens when we have an *even* number of samples. We define $\vec{s} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$, which has $N = 6$ samples. **Compute the DFT coefficients $\vec{S}$ for signal $\vec{s}$.**

Hint: Write $\vec{s}$ as $a + b\cos\left(\frac{2\pi}{6}pn\right)$ for some constants a, b, p, and use the fact that this signal has period 2.

Solution: We have $N = 6$ samples, which we will denote $n \in \{0, 1, \ldots, 5\}$. The signal repeats after a period $T = 2$, so the cosine function should also have period 2. This will occur when $p = 3$. Then to account for the scaling and shifting of the cosine from $[-1, 1]$ to $[0, 1]$, we need $a = \frac{1}{2}, b = \frac{1}{2}$. This gives

$$s[n] = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2\pi}{6}(3)n\right) \tag{22}$$

We can decompose $s[n]$ as

$$s[n] = \frac{1}{2}e^{(0)n} + \frac{1}{4}e^{-j\frac{2\pi(3)}{6}n} + \frac{1}{4}e^{j\frac{2\pi(3)}{6}n} \tag{23}$$

$$\vec{s} = \frac{1}{2}\sqrt{N}\vec{u_0} + \frac{1}{4}(\sqrt{N}\vec{u_3} + \sqrt{N}\,\overline{\vec{u}_3}) \tag{24}$$

$$= \frac{\sqrt{6}}{2}\vec{u_0} + \frac{\sqrt{6}}{4}(\vec{u_3} + \vec{u}_{6-3}) \tag{25}$$

$$= \frac{\sqrt{6}}{2}\vec{u_0} + \frac{\sqrt{6}}{2}\vec{u_3}. \tag{26}$$

From part (b), $\vec{u}_3 = \overline{\vec{u}_3}$. Therefore both of the $\vec{u}_3$ and $\overline{\vec{u}_3}$ terms are added together and contribute to the $k = \frac{N}{2} = 3$rd term of $\vec{S}$. Additionally, we have a term from $k = 0$ because of the constant offset of the signal with 0 frequency. Thus,

$$S[k] = \begin{cases} \frac{\sqrt{6}}{2} & k = 0, 3. \\ 0 & k \neq 0, 3. \end{cases} \tag{27}$$

Having $\vec{u}_3 = \overline{\vec{u}_3}$ shows us an interesting result of using an *even* number of samples. When N is even, the $\vec{u}_{\frac{N}{2}}$ column will be entirely real because

$$u_{\frac{N}{2}}[l] = e^{-j\frac{2\pi}{N}\frac{N}{2}l} = e^{-j\pi l},$$

and so the entries will alternate between -1 and 1.

6. Survey

Please fill out the survey here. As long as you submit it, you will get credit. Thank you!

7. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

Contributors:

- Kourosh Hakhamaneshi.
- Kuan-Yun Lee.
- Nathan Lambert.
- Sidney Buchbinder.
- Gaoyue Zhou.
- Anant Sahai.
- Ashwin Vangipuram.
- John Maidens.
- Geoffrey Négiar.
- Yen-Sheng Ho.
- Harrison Wang.
- Regina Eckert.