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.

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** $\frac{d}{d\vec{w}} c_i(\vec{w}_*)$.

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* $\vec{x}_i^\top \vec{w} = \sum_{j=1}^n x_i[j]w[j] = x_i[g]w[g] + \sum_{j \neq g} x_i[j]w[j]$*.)*

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

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

(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 $\frac{d^2}{d\vec{w}^2}c_i(\vec{w}_*)$.

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* $\vec{x}_i^\top \vec{w} = \sum_{j=1}^n x_i[j]w[j] = x_i[h]w[h] + \sum_{j\neq h} x_i[j]w[j]$.*)*

iii) The expression in part (ii) is for the $[g, h]$-th component of the second derivative. $\frac{1}{2}$ times this times $\vec{\delta w}[g]$ times $\vec{\delta w}[h]$ 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?**

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

(c) Now we have successfully expressed the second order approximation of $c_i(\vec{w}_* + \vec{\delta w})$. 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** $c_{total}(\vec{w}_* + \vec{\delta w})$ **using results from (a) and (b)?**

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

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**

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

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**?**

(e) Consider a least squares problem:

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

Show that:

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

where

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

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

(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: $c_{logistic}^+(p) = \ln\big(1+e^{-p}\big),\ c_{logistic}^-(p) = \ln(1+e^p)$.
Compute the first and second derivatives of the above expressions with respect to p.

(g) Run the Jupyter Notebook and answer the following questions.
i) **In Example 2, why does mean classification fail?**
ii) **In Example 3, for what data distributions does ordinary least squares 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?**
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?

(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.**

(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 $(AB)^* = \overline{(AB)^T} = \overline{B^T A^T} = \overline{B^T}\ \overline{A^T} = B^*A^*$

(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.

(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 $\mathbb{C}^n$.**)

(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 $M^* = M$.

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**

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

(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 $\omega_N = e^{j\frac{2\pi}{N}}$.

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

(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?**

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

(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.**

(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.

(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?**

(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?

5. Discrete Fourer 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.

$$\vec{x} = \begin{bmatrix} x[0] & \dots & x[N-1] \end{bmatrix}^T \tag{5}$$

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}?**

(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 $k = 1, 2, \dots, N-1$.**

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

(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 $y_-[n] = e^{-j\frac{2\pi n}{7}}$.**

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

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

(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.

(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 $\vec{x}_2$.**

(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 $\vec{X}_2$.**

(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.**

(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.

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.