EE127/227A Homework 1

Vighnesh Iyer

September 5, 2018

Exercise 1 (CVX Installation)

For Python users: Install the package CVXOPT and run the tutorial on Linear Program solvers: `http://cvxopt.org/examples/tutorial/lp.html`
For MATLAB users: Install CVX and make sure you can run the least squares code: `http://cvxr.com/cvx/doc/quickstart.html`

Solution:

I'm using CVXOPT. From Wikipedia, linear programs can be expressed in canonical form:

$$\begin{aligned} &\text{Minimize } \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to } A\boldsymbol{x} \le \boldsymbol{b} \\ &\text{and } \boldsymbol{x} \ge 0 \end{aligned}$$

In the tutorial, the following problem is given:

$$\begin{aligned} \text{Minimize } & 2x_1 + x_2 \\ \text{subject to } & -x_1 + x_2 \le 1 \\ & x_1 + x_2 \ge 2 \\ & x_2 \ge 0 \\ & x_1 - 2x_2 \le 4 \end{aligned}$$

So we can write down the matrices by pattern matching:

$$c = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 4 \end{bmatrix}$$

$$A = \begin{bmatrix} -1 & 1 \\ -1 & -1 \\ 0 & -1 \\ 1 & -2 \end{bmatrix}$$

Then use cvxopt to solve

```python
from cvxopt import matrix, solvers
A = matrix([ [-1.0, -1.0, 0.0, 1.0], [1.0, -1.0, -1.0, -2.0] ])
```

```
b = matrix([ 1.0, -2.0, 0.0, 4.0 ])
c = matrix([ 2.0, 1.0 ])
sol=solvers.lp(c,A,b)
print(sol['x'])
# [ 5.00e-01]
# [ 1.50e+00]
```

Exercise 2 (About general optimization)

In this exercise, we test your understanding of the general framework of optimization and its language. We consider an optimization problem in standard form:

$$p^* = \min_{x \in \mathbb{R}^n} \ f_0(x) \ : \ f_i(x) \leq 0, \ \ i = 1, \ldots, m.$$

In the following we denote by $\mathcal{X}$ the feasible set. For the following statements, provide a proof or counter-example.

1. Any optimization problem can be expressed as one with a linear objective.
2. Any optimization problem can be expressed as one without any constraints.
3. Any optimization problem can be recast as a linear program, provided one allows for an infinite number of constraints.
4. If one inequality is strict at the optimum, then we can remove it from the original problem and obtain the same solution.
5. If the problem involves the minimization over more than one variable, say y and x, then we can exchange the minimization order without altering the optimal value:
$$\min_x \min_y \ F_0(x, y) = \min_y \min_x \ F_0(x, y)$$
6. If the problem involves the minimization of an objective function of the form
$$f_0(x) = \max_y \ F_0(x, y),$$
then $p^* \geq d^*$, where
$$p^* := \min_x \max_y F_0(x, y)$$
$$d^* := \max_y \ \min_x \ F_0(x, y).$$
Hint: consider the function $y \to \min_{x'} \ F_0(x', y)$ and a similar function of x.

Solution:

1. Yes, this is true. Move the non-linear objective function to the constraint list as such:
$$p^* = \min_y y$$
$$\text{subject to } f_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m$$
$$\text{and } f_0(\boldsymbol{x}) \leq y$$
$$\text{for } \boldsymbol{x} \in \mathbb{R}^n$$
2. Yes, this is true. Move the constraints to the original objective function to increase the cost when the constraint isn't met.
$$p^*_{new} = \min_{x \in \mathbb{R}^n} f_0(x) + \sum_{i=1}^{m} \max(f_i(x), 0)$$

3. Yes, this is true. First, from subsection 1, we can always transform an optimization problem into one with a linear objective. Then, to express any (potentially non-linear) constraint as a set of infinite linear constraints, form a bijection $\mathbb{R}^n \to \mathbb{R}$ (e.g. decimal place interleaving in the interval (0, 1)), evaluate the constraint on all points in $\mathbb{R}^n$ and for each point create a row in A and entry in b that indicates whether the constraint is satisfied or not.

4. A counter-example exists. An inequality being strict at the optimum means that it is **not** satisfied at the optimum, but just barely so.

$$\begin{aligned} x^* &= \min_x x^2 : x > 0 \\ \text{without strict constraint } x^* &= 0 \\ \text{with strict constraint } x^* &= \text{infinitely close to 0, but not 0} \end{aligned}$$

5. Yes, this is true. A minimization over 1 variable must consider all possible values of the other variable, so the order of minimization cannot matter. If minimization over 1 variable can't be performed without knowledge of the other variable, then the order of minimization cannot matter, and in the case where minimization over 1 variable allows for elimination of that variable the same can be said.

6. **TODO TODO TODO**

Exercise 3 (1D Convolution)

A 1D convolutional filter takes an input sequence $x = (x_t)_{t\in\mathbb{Z}}$ and a finite sequence $h = (h_t)_{t\in[1,m]}$ and produces an output sequence according to the rule

$$y_t = h_1 x_t + h_2 x_{t-1} + ... + h_m x_{t-m+1}, t \in \mathbb{Z}$$

This operation is mathematically denoted $y = h * x$ and called a convolution. Now consider a sequence x such that $x_t = 0$ for $t < 1$ and $t > n$. We will from now refer to *infinite-dimensional* vector x as the *finite-dimensional* vector $(x_1, ..., x_n)$.

1. Show that the output sequence $(y_t)_{t\in\mathbb{Z}}$ is zero outside a band. In particular, you will show that $y_t = 0$ for $t \leq a$ and $t > b$ for (a, b) that you will determine as a function of n and m. The output sequence can therefore be represented as a $(b - a)$-dimensional vector that we will note $y = (y_{a+1}, ..., y_b)$.
2. Express the relationship between the output vector y and x as $y = T(h) \cdot x$, with $T(h)$ a matrix of shape $(b - a, n)$. What structure does that matrix have?
3. We consider two n-th order polynomials p, q:

$$\begin{aligned} p(s) &= p_0 s^0 + p_1 s^1 + ... + p_n s^n \\ q(s) &= q_0 s^0 + q_1 s^1 + ... + q_n s^n. \end{aligned}$$

   Express the product of the polynomials in terms of vectors of coefficients of p, q, and powers of s, and a certain Toeplitz matrix, which you will determine.
4. A given T-vector r gives the daily rainfall in some region over a period of T days. The vector h gives the daily height of a river in the region (above its normal height). By careful modeling of water flow, or by fitting a model to past data, it is found that these vectors are (approximately) related by convolution: $h = g * r$, where $g = (0.1, 0.4, 0.5, 0.2)$. How many days after a one day heavy rainfall is the river height most affected? And, how many days does it takes for the river height to return to the normal height once the rain stops?

Solution:

1. We can write y_t as:

$$y_t = \begin{cases} 0 & t \leq 0 \\ \sum_{i=1}^{m} h_i x_{t-i-1} & t > 0 \wedge t \leq n + m - 1 \\ 0 & t > n + m - 1 \end{cases}$$
$$a = 0$$
$$b = n + m - 1$$

   When $t \leq 0$, all the terms of y_t become 0. After that point, every value of y_t can be non-zero until the x_t vector has 'passed' the convolution taps.

2. The matrix is a Toeplitz matrix (each descending diagonal from left to right is constant).

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{b-a} \end{bmatrix} = \begin{bmatrix} h_1 & 0 & \dots & 0 & 0 \\ h_2 & h_1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & h_1 & 0 \\ h_m & h_{m-1} & \dots & h_2 & h_1 \\ 0 & h_m & \dots & h_{m-2} & h_{m-1} \\ 0 & 0 & \dots & h_{m-1} & h_{m-2} \\ 0 & 0 & 0 & h_m & h_{m-1} \\ 0 & 0 & 0 & 0 & h_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

3. The product of $p(s)$ and $q(s)$ can be given in the form $(\mathbf{Tp})^T\mathbf{s}$, where $\mathbf{T}$ is the constructed Toeplitz matrix, $\mathbf{p}$ is a vector composed of polynomial coefficients (i.e. $p_0, p_1, \dots, p_n$) , and $\mathbf{s}$ is a vector of powers of s (i.e. $s^0, s^1, \dots, s^n$).

$$T = \begin{bmatrix} p_0 & 0 & \dots & 0 & 0 \\ p_1 & p_0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & p_1 & 0 \\ p_n & p_{n-1} & \dots & p_1 & p_0 \\ 0 & p_n & \dots & p_{n-2} & p_{n-1} \\ 0 & 0 & \dots & p_{n-1} & p_{n-2} \\ 0 & 0 & 0 & p_n & p_{n-1} \\ 0 & 0 & 0 & 0 & p_n \end{bmatrix}$$

4. It takes 2 days after heavy rainfall for the river height to become maximal because there are 2 taps of 'delay' before the heavy rainfall day lands on the 0.5 multiplier. Similarly, it takes 3 days after heavy rainfall (counting the day that the rain stops) for the river height to go back to 'normal' because there are only 4 coefficients in g before the heavy rainfall day no longer affects h.

Exercise 4 (Norm and angles)

1. Let $x, y \in \mathbb{R}^n$ be two unit-norm vectors, that is, such that $\|x\|_2 = \|y\|_2 = 1$. Show that the vectors $x - y$ and $x + y$ are orthogonal. Draw on the 2D plane the two vectors and all the necessary shapes to graphically solve the exercise. You may use right angles, circles and straight lines to make your point.

2. Show that the following inequalities hold for any vector $x \in \mathbb{R}^n$:
$$\frac{1}{\sqrt{n}}\|x\|_2 \leq \|x\|_\infty \leq \|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2 \leq n\|x\|_\infty.$$
Hint: think about using Cauchy-Schwarz's inequality, and instantiate it on the corresponding norms.

3. Show that for any non-zero vector x,
$$\text{card}(x) \geq \frac{\|x\|_1^2}{\|x\|_2^2},$$
where card(x) is the *cardinality* of the vector x, defined as the number of non-zero elements in x. Find all vectors x for which the lower bound is attained.

Solution:

1.
$$\begin{aligned}
x - y \text{ orthogonal to } x + y \rightarrow (x-y)\cdot(x+y) &= 0\\
(x-y)_i &= x_i - y_i\\
(x+y)_i &= x_i + y_i\\
(x-y)\cdot(x+y) &= \sum_{i=1}^{n}(x-y)_i(x+y)_i = \sum_{i=1}^{n}(x_i - y_i)(x_i + y_i)\\
&= \sum_{i=1}^{n} x_i^2 - y_i^2 = \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i^2\\
&= ||x||_2^2 - ||y||_2^2 = 1 - 1 = 0
\end{aligned}$$

TODO TODO TODO 2D DRAWING

2. Prove the inequalities from left to right

$$\text{Prove: } \frac{1}{\sqrt{n}}||x||_2 \leq ||x||_\infty$$
$$||x||_2^2 = \sum_{i=1}^{n} x_i^2 \leq n \cdot \max_{i=1,\dots,n} x_i^2 n \cdot \max_{i=1,\dots,n} |x_i|^2 = n||x||_\infty^2$$
$$\rightarrow ||x||_2^2 \leq n||x||_\infty^2$$
$$\rightarrow \frac{1}{\sqrt{n}}||x||_2 \leq ||x||_\infty$$

$$\text{Prove: } \|x\|_\infty \leq \|x\|_2$$
$$\|x\|_\infty = \max_{i=i,\dots,n} |x_i| = \max_{i=1,\dots,n} \sqrt{|x_i|^2} \leq \sqrt{\sum_{i=1}^{n} |x_i|^2} = \|x\|_2$$
$$\rightarrow \|x\|_\infty \leq \|x\|_x$$

$$\begin{aligned}
\text{Prove: } &\|x\|_2 \leq \|x\|_1 \\
\text{In general } &\sqrt{a_1 + a_2 + \cdots + a_n} \leq \sqrt{a_1} + \sqrt{a_2} + \cdots + \sqrt{a_n} \\
&\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} \leq \sum_{i=1}^{n} \sqrt{|x_i|^2} = \|x\|_1 \\
&\rightarrow \|x\|_2 \leq \|x\|_1
\end{aligned}$$

$$\begin{aligned}
\text{Prove: } &\|x\|_1 \leq \sqrt{n}\|x\|_2 \\
\text{Let: } &x = \begin{bmatrix} |x_1| & |x_2| & \dots & |x_n| \end{bmatrix}^T \\
&y = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \\
\text{Applying C-S: } &|x^T y| \leq \|x\|_2 \|y\|_2 \\
&|\sum_{i=1}^{n} |x_i|| \leq \|x\|_2 \sqrt{n} \\
&\|x\|_1 \leq \sqrt{n}\|x\|_2
\end{aligned}$$

$$\begin{aligned}
\text{Prove: } &\sqrt{n}\|x\|_2 \leq n\|x\|_\infty \\
\text{We had shown: } &\|x\|_2^2 \leq n\|x\|_\infty^2 \\
&n\|x\|_2^2 \leq n^2\|x\|_\infty^2 \\
&\sqrt{n}\|x\|_2 \leq n\|x\|_\infty
\end{aligned}$$

3. **TODO TODO TODO**

Exercise 5 (Comparing text)

In this exercise, we use the word-frequency vector representation of text for comparing text documents. For mathematical modeling of transforming text into vectors, refers to Lecture 2, Slide 6 (Example 1: bag of words). In this context, similarity between two documents may be measured by means of the angle θ between the two frequency vectors representing the documents, the documents being maximally “different” when the corresponding frequency vectors are orthogonal. Consider the following headlines from the web edition of the New York Times on Dec. 7, 2010:

(a) Suit Over Targeted Killing in Terror Case Is Dismissed. A federal judge on Tuesday dismissed a lawsuit that sought to block the United States from attempting to kill an American citizen, Anwar Al-Awlaki, who has been accused of aiding Al Qaeda.

(b) In Tax Deal With G.O.P., a Portent for the Next 2 Years. President Obama made clear that he was willing to alienate his liberal base in the interest of compromise. Tax Deal Suggests New Path for Obama. President Obama agreed to a tentative deal to extend the Bush tax cuts, part of a package to keep jobless aid and cut payroll taxes.

(c) Obama Urges China to Check North Koreans. In a frank discussion, President Obama urged China’s president to put the North Korean government on a tighter leash after a series of provocations.

(d) Top Test Scores From Shanghai Stun Educators. With China’s debut in international standardized testing, Shanghai students have surprised experts by outscoring counterparts in dozens of other countries.

1. First simplify the text (remove plurals, convert verb to infinite tense, etc.) and then find the frequency vectors representing the text against the dictionary $V = \{\text{aid, kill, deal, president, tax, china}\}$.
2. Compare the four pieces of text by computing the cosine distance (i.e the cosine of the angles) between any pairs of texts.

Solution:

1.

$$a = \begin{bmatrix}1 & 2 & 0 & 0 & 0 & 0\end{bmatrix}^T$$
$$b = \begin{bmatrix}1 & 0 & 3 & 2 & 4 & 0\end{bmatrix}^T$$
$$c = \begin{bmatrix}0 & 0 & 0 & 2 & 0 & 2\end{bmatrix}^T$$
$$d = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}^T$$

2.
```
pairs = [
    (a,a),(a,b),(a,c),(a,d),
    (b,b),(b,c),(b,d),(c,c),(c,d),(d,d)
]
pairs_eng = [
    '(a,a)','(a,b)','(a,c)','(a,d)',
    '(b,b)', '(b,c)','(b,d)','(c,c)','(c,d)','(d,d)'
]
for ((x,y),eng) in zip(pairs,pairs_eng):
    print(eng + ":")
    print(np.dot(x.T, y) / (np.linalg.norm(x,2) * np.linalg.norm(y,2)))
```

```
(a,a):
0.9999999999999998
(a,b):
0.0816496580927726
(a,c):
0.0
(a,d):
0.0
(b,b):
1.0
(b,c):
0.2581988897471611
(b,d):
0.0
(c,c):
0.9999999999999998
(c,d):
0.7071067811865475
(d,d):
1.0
```

Exercise 6 (Linear functions and projections)

1. Suppose x is an n-vector, with $n = 2m - 1$ and $m \geq 1$. We define the middle element value of x as x_m. Define
$$f(x) = x_m - \frac{1}{n}\sum_{i=1}^{n} x_i$$
which is the difference between the middle element value and the average of the coefficients in x. Express f in the form $f(x) = a^T x$, where a is an n-vector.

2. Now let b and x be vectors of $\mathbb{R}^2$ (with $\|b\|_2 = 1$). Draw on the 2-D plane b, x and show the value of $b^T x$ on the graph.

Now we will focus on Senator Voting data. This data provides information about senator vote x and senator political affiliation y. We provide you with four different vectors (a_1, a_2, a_3, a_4) precomputed by the EECS127 staff. Each of these vectors can be used to define a linear function $f_a : x \to a^T x$.

3. Load the files in *senator.zip*, and let X be the data matrix (with each row a senator, and each column a bill). Center the data matrix X (which is a standard preprocessing step for data analysis) and then compute for each vector a the score of each senator. We provide you with the skeleton code to load the data and visualize the scores in contrast to the affiliation y. You will compute such plots using the existing code and explain them.

4. Which direction among the vectors a_1, a_2, a_3, a_4 do you prefer in order to produce a score that effectively summarizes the affiliation? Justify your answer.

Solution:

1. Construct **a** as such:
$$a = \begin{bmatrix} -\frac{1}{n} & -\frac{1}{n} & \dots & -\frac{1}{n}+1 & \dots & -\frac{1}{n} & -\frac{1}{n} \end{bmatrix}^T$$

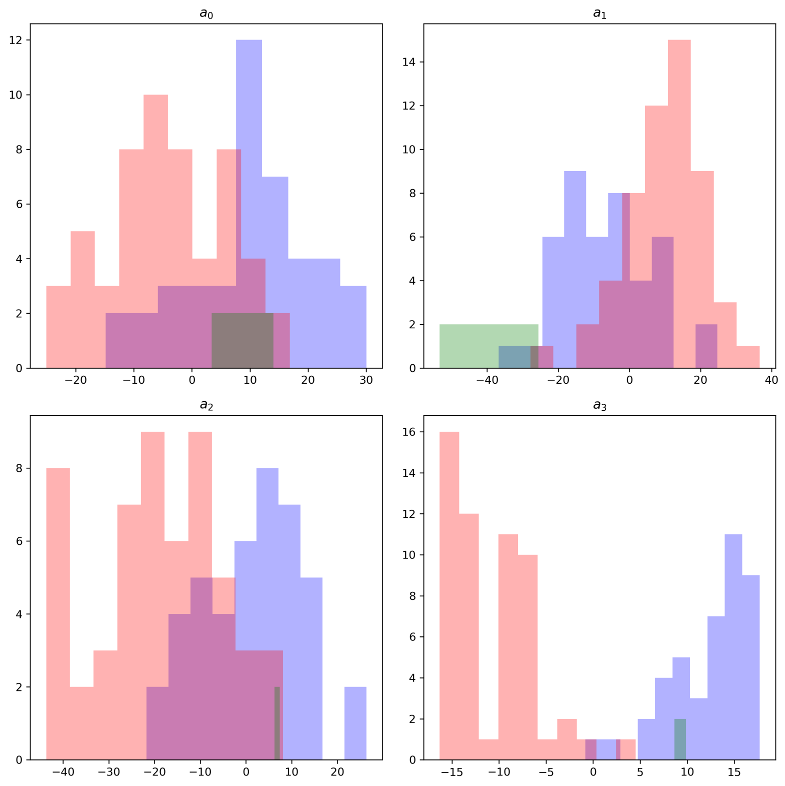
2. **TODO TODO TODO Include drawing which is written in notes**

3. To subtract the mean from the data matrix X:

```
X_bar = np.zeros(X.shape)
col_means = X.mean(axis=0)
for col in range(X.shape[1]):
    for row in range(X.shape[0]):
        X_bar[row,col] = X[row,col] - col_means[col]
print(X_bar)
```

To compute the senator scores:

```
senator_scores = np.dot(a_vectors, X_bar)
```



4. Direction a_4 clearly works the best to separate senators based on affilation. Look at the large score difference between red and blue senators for vector a_4 but the large overlap with the other vectors.

Exercise 7 (Customer purchase history matrix)

A store keeps track of its sales of products from K different product categories to N customers over some time period, like one month. K might be on the order of 1000 and N might be 100000. The data is stored in an $N \times K$ matrix C, with C_{ij} being the total dollar purchases of product j by customer i All the entries of C are non-negative. The matrix C is typically sparse, i.e., many of its entries are zero.

1. What is $C\mathbf{1}$? ($\mathbf{1}$ is a K-vector of 1s)
2. What is $C^T\mathbf{1}$? ($\mathbf{1}$ is a N-vector of 1s)
3. Give a short matrix-vector expression for the total dollar amount of all purchases, by all customers.
4. What does it mean if $(CC^T)_{kl} = 0$? Your answer should be simple English.

We are now interested in efficient computations. In our setting, note that the data matrix C is large but very sparse. The number of non zero-valued elements divided by the total number of elements is called the density d of the matrix C. Let w in $\mathbb{R}^K$ be a given weighting vector. Assume that we center the rows (removing the average row to every row), obtaining a new row-centered matrix C_m.

5. Provide an explanation on how to efficiently compute the matrix vector product $v = C_m w$.
6. Mathematically estimate the gain speed of your method vs the naive method (of forming C_m first) as a function of K, N, and d.
7. Confirm your findings with an empirical study: we provide you with a matrix C and a vector w in the Jupyter Notebook matrix-vector.ipynb. Using the skeleton code, compare the sparse matrix method against the naive method. Comment on the time improvement and attach your notebook here in pdf format for grading. Let us note that we do not ask the students to create their own sparse matrix functionalities.
8. Assume that we add one row (data point). Explain how to efficiently update the resulting vector v accordingly.

Solution:

1. $C\mathbf{1}$ returns a vector representing the total purchase amount (in dollars) per customer.
2. $C^T\mathbf{1}$ returns a vector representing the total purchased amount (in dollars) per product.
3. $(C\mathbf{1}_K)^T\mathbf{1}_N$ where $\mathbf{1}_x$ is a length x vector of 1s.
4. If $(CC^T)_{k,l} = 0$ it means customer k and customer l didn't purchase any products in common.
5. Let $\mathbf{m}$ denote the row average vector, which containes K elements. We can express the first row (sum) of $\mathbf{v} = C_m\mathbf{w}$ as such:

$$\begin{aligned} \mathbf{v}_0 &= (C_{0,0} - m_0)w_0 + (C_{0,1} - m_1)w_1 + \cdots + (C_{0,K-1} - m_{K-1})w_{K-1} \\ \text{Combining terms: } \mathbf{v}_0 &= (C_{0,0}w_0 + \cdots + C_{0,K-1}w_{K-1}) - (m_0w_0 + \cdots + m_{K-1}w_{K-1}) \end{aligned}$$

   Note that the first term can be formed with $C_{sparse} \cdot \mathbf{w}$ and the second term can be formed with $\mathbf{m} \cdot \mathbf{w}^T$ and they can be subtracted.

6. The naive method would first perform $N \cdot K$ subtractions to form C_m, then $K \cdot N$ multiplications and $(K-1) \cdot N$ additions to form $\mathbf{v}$.

   Total naive: $2NK$ multiplications and $(K-1)N$ additions.

   The optimized method would perform K multiplications and $K-1$ additions to form $\mathbf{m} \cdot \mathbf{w}^T$. Due to sparsity, the operations used to compute $C_{sparse} \cdot \mathbf{w}$ will be scaled from the naive method by density d to get $K \cdot N \cdot d$ multiplications and $(K-1) \cdot N \cdot d$ additions. Finally, to form $\mathbf{v}$, we need N subtractions.

   Total optimized: $K + KNd$ multiplications and $K - 1 + (K-1)Nd + N$ add/sub operations.

   Rounding $K-1$ to K and estimating that multiplications have about the same throughput as add or subtract ops:

$$\begin{aligned} \text{Naive ops } &= 3NK \\ \text{Opt ops } &= 2K + 2KNd + N \end{aligned}$$

7. In this notebook, $N = K = 10000$ and $d = 0.1$, so the naive ops $= 300000000$, and opt ops $= 20030000$ for an estimated 15X speedup.

   The naive implementation:

```
(C - r_avg) @ w

CPU times: user 487 ms, sys: 211 ms, total: 697 ms
Wall time: 472 ms
```

   The optimized implementation:

```
np.subtract(C_sparse @ w, r_avg @ w)

CPU times: user 27.9 ms, sys: 26.8 ms, total: 54.7 ms
Wall time: 53 ms
```

   Equivalence Check:

```
gold = (C - r_avg) @ w
opt = np.subtract(C_sparse @ w, r_avg @ w)
np.allclose(gold, opt) # account for floating-point errors

True
```

   I actually got a 9X speedup, which is close enough. Probably I need to put more weight on the multiplication ops over the add/sub ops.

8. Adding a new row will change $\mathbf{m}$, the row average vector. Assuming $C_{sparse} \cdot \mathbf{w}$ is cached, just recompute $\mathbf{m} \cdot \mathbf{w}^T$ and subtract to refresh the existing elements of $\mathbf{v}$.

   Then to add a new element to $\mathbf{v}$, take the new row $\mathbf{r}$ on its own and perform $\mathbf{r} \cdot \mathbf{w}$, then subtract $\mathbf{m} \cdot \mathbf{w}^T$ from it.