Foundations of Data Science and Machine Learning – *Homework 2*

Isaac Martin

Last compiled February 13, 2023

EXERCISE 1. In d-dimensions there are at most d unit vectors that are pairwise orthogonal. However, if you wanted a set of vectors that were "almost orthogonal" you might squeeze in a few more. For example, in 2-dimensions if you wanted vectors at least 45 degrees apart, then you could fit in three vectors. Suppose you wanted to find 1000 almost orthogonal vectors in 100 dimensions. Here are two ways you could try doing it:

(a) Begin with 1000 orthonormal 1000-dimensional vectors, and then project them to a random 100-dimensional space.

(b) Generate 1000 100-dimensional random standard Gaussian vectors.

Solution:

(a) We generated 1000 1000-dimensional standard Gaussian vectors, stuck these in the columns of a matrix A and then performed QR-factorization to extract out the orthonormal basis for the span of A. The probability of choosing 1000 vectors which are linearly dependent in $\mathbb{R}^{1000}$ ought to be zero – if you have chosen $k < 1000$ vectors already then the k-dimensional subspace spanned by these vectors has measure zero in $\mathbb{R}^n$ – but we check that the determinant of Q is not zero before proceeding.

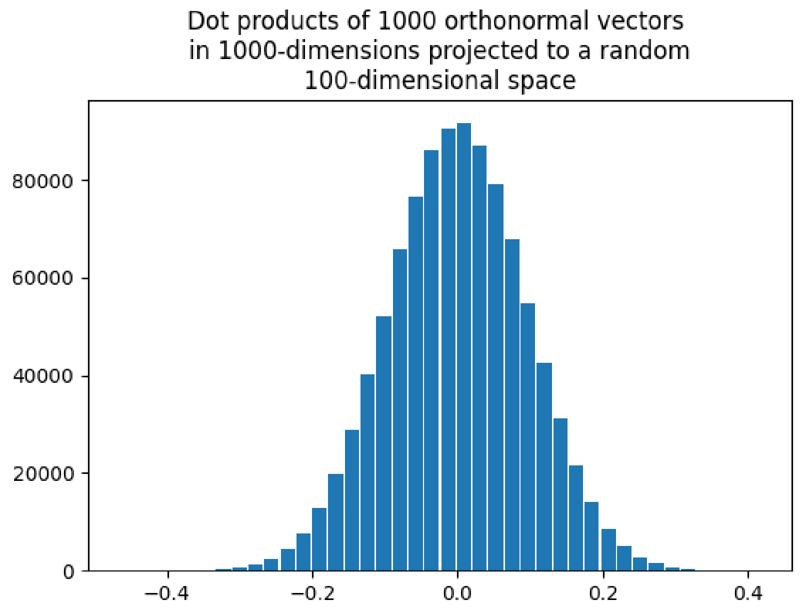
As an aside: rather than fixing a random 100-dimensional subspace, we simply project the columns of Q to the subspace spanned by $e_1, ..., e_{100}$. This is actually equivalent to projecting to a random 100-dimensional subspace. Any two d-dimensional linear subspaces of $\mathbb{R}^n$ are related by the action of $O(n)$, that is, if $V, W \subseteq \mathbb{R}^n$ are linear subspaces then there is some $P \in O(n)$ such that $P \cdot V = W$. Furthermore, P is an isometry of $\mathbb{R}^n$, meaning for any $u, v \in \mathbb{R}^n$, $\langle u, v\rangle = \langle Pu, Pv\rangle$. We're using "the closeness of the dot product to zero" as our measure of "orthogonal-ness", so we might as well take the dot product in $\operatorname{span}_{\mathbb{R}}\{e_1, ..., e_{100}\}$ to make the projection easier.

We now project the columns of Q to the subspace spanned by $e_1, ..., e_{100}$, i.e. we truncate the 900 tail components from each column of Q. At this point, we would like to take the pairwise dot product of distinct columns and examine the results. However, the projection $\mathbb{R}^{1000} \longrightarrow \mathbb{R}^{100}$ results in a large loss of norm, and because we will be comparing this process to Gaussian sampling in part (b), we renormalize our vectors to ensure that the magnitude of our dot product solely reflects the angle between the vectors.

The results of this process can be seen in the "discussion" section.

(b) For method b, we generated 1000 100-dimensiona iid standard Gaussian vectors and normalized them before calculating the pairwise dot products of distinct vectors. This was more straightforward than method 2 as it doesn't involve projecting to a lower dimensional vector space.

Discussion. Here are histograms illustrating the results methods (a) and (b).



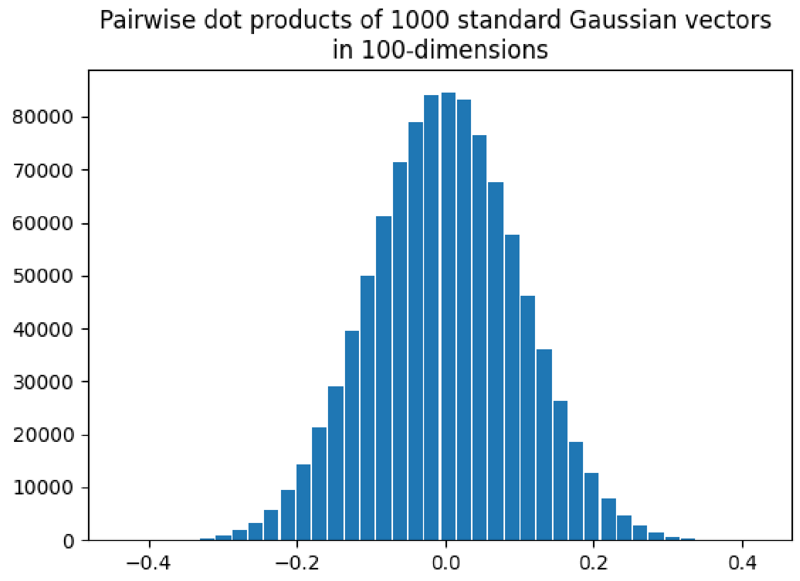


Figure 1: Density of pairwise dot product values for the two "almost orthogonal" methods

As you can see, the two methods produced similar results. Neither produced dot product values outside of $[-0.35, 0.35]$, and hence no two vectors were closer than $\cos^{-1}(0.35) = 70^\circ$ apart. The second method was marginally worse, as it has a slightly flatter density curve. $\square$

Exercise 2.

(a) Fix $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ both with an ℓ_2-norm of 1. Suppose that $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}^r$ is a linear map satisfying

$$(1-\varepsilon)\|\mathbf{x}+\mathbf{y}\|_2^2 \leq \|\Phi(\mathbf{x}+\mathbf{y})\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}+\mathbf{y}\|_2^2 \tag{1}$$

and

$$(1-\varepsilon)\|\mathbf{x}-\mathbf{y}\|_2^2 \leq \|\Phi(\mathbf{x}-\mathbf{y})\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}-\mathbf{y}\|_2^2. \tag{2}$$

Use the identity $4\langle \mathbf{u}, \mathbf{v}\rangle = \|\mathbf{u}+\mathbf{v}\|_2^2 - \|\mathbf{u}-\mathbf{v}\|_2^2$ to show that

$$|\langle \mathbf{x}, \mathbf{y}\rangle - \langle \Phi(\mathbf{x}), \Phi(\mathbf{y})\rangle| \leq \varepsilon \|\mathbf{x}\|_2 \|\mathbf{y}\|_2.$$

(b) If $\chi = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ is an arbitrary fixed set in $\mathbb{R}^d$ and $\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}^r$ is a random matrix with independent mean-zero variance $1/r$ Gaussian entries, how should the embedding dimension r scale in terms of n, d, ε so that with probability at least 0.9 it holds

$$|\langle \mathbf{x}_j, \mathbf{x}_k\rangle - \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_k)\rangle| \leq \varepsilon \|\mathbf{x}_j\|_2 \|\mathbf{x}_k\|_2.$$

Proof:

(a) Rearrange both equation (1) and (2) as follows:

$$-\varepsilon\|\mathbf{x}+\mathbf{y}\|_2^2 \leq \|\Phi(\mathbf{x}+\mathbf{y})\|_2^2 - \|\mathbf{x}+\mathbf{y}\|_2^2 \leq \varepsilon\|\mathbf{x}+\mathbf{y}\|_2^2 \tag{3}$$

$$-\varepsilon\|\mathbf{x}-\mathbf{y}\|_2^2 \leq \|\Phi(\mathbf{x}-\mathbf{y})\|_2^2 - \|\mathbf{x}-\mathbf{y}\|_2^2 \leq \varepsilon\|\mathbf{x}-\mathbf{y}\|_2^2. \tag{4}$$

Adding equation (4) to the negative of (3) yields

$$\begin{aligned}-\varepsilon(\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2) \leq \|\mathbf{x}+\mathbf{y}\|_2^2 - \|\mathbf{x}-\mathbf{y}\|_2^2 - (\|\Phi(x+y)\|_2^2 - \|\Phi(x-y)\|_2^2)\\ \leq \varepsilon(\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2).\end{aligned}$$

Let's examine the term $\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2$. Consider the following illustration:

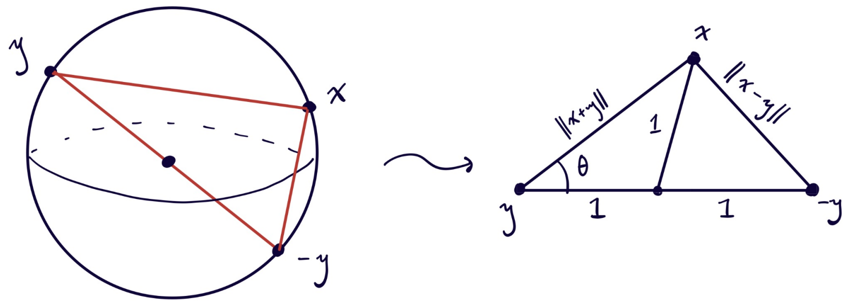


Figure 2: The points $\mathbf{x}$ and $\mathbf{y}$ define a triangle

Since $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$, x and y lie on the unit sphere in $\mathbb{R}^d$, and the sum $\|\mathbf{x}+\mathbf{y}\|_2 + \|\mathbf{x}-\mathbf{y}\|_2$ can be thought of as a portion of the perimeter of the triangle in Figure (2). This perimeter is maximized when $\theta = \pi/4$, in which case $\|\mathbf{x}+\mathbf{y}\| = \|\mathbf{x}-\mathbf{y}\| = \sqrt{2}$. Because

$$\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2 = (\|\mathbf{x}+\mathbf{y}\| + \|\mathbf{x}-\mathbf{y}\|)^2 - 2\|\mathbf{x}+\mathbf{y}\|\|\mathbf{x}-\mathbf{y}\|,$$

one can show that $\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2$ attains a maximum value on the unit sphere whenever either $\|\mathbf{x}+\mathbf{y}\| + \|\mathbf{x}-\mathbf{y}\|$ is maximized or when $\mathbf{x}=\mathbf{y}$. In either case, $\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2 = 4$. Combining this fact with the inequality above, we get that

$$\begin{aligned} -4\varepsilon &\le -\varepsilon(\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2) \\ &\le \|\mathbf{x}+\mathbf{y}\|_2^2 - \|\mathbf{x}-\mathbf{y}\|_2^2 - (\|\Phi(x+y)\|_2^2 - \|\Phi(x-y)\|_2^2) \\ &\le \varepsilon(\|\mathbf{x}+\mathbf{y}\|_2^2 + \|\mathbf{x}-\mathbf{y}\|_2^2) \le 4\varepsilon, \end{aligned}$$

or cutting out the 2nd and 4th terms,

$$-4\varepsilon \le \|\mathbf{x}+\mathbf{y}\|_2^2 - \|\mathbf{x}-\mathbf{y}\|_2^2 - (\|\Phi(x+y)\|_2^2 - \|\Phi(x-y)\|_2^2) \le 4\varepsilon.$$

We can now use the linearity of Φ together with the given identity to see

$$-4\varepsilon \le 4\langle \mathbf{x},\mathbf{y}\rangle - 4\langle \Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle \le 4\varepsilon,$$

and so we have

$$\big|\langle \mathbf{x},\mathbf{y}\rangle - \langle \Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle\big| \le \varepsilon = \varepsilon\|\mathbf{x}\|_2\|\mathbf{y}\|_2$$

as desired.

(b) Denote by $\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ the normalization of a vector $\mathbf{x}\in\mathbb{R}^d$. By the bilinearity of the inner product and the linearity of Φ, we can factor out the product of norms:

$$\begin{aligned} \big|\langle \mathbf{x},\mathbf{y}\rangle - \langle \Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle\big| &= \big|\langle\ \|\mathbf{x}\|\cdot\hat{\mathbf{x}}, \|\mathbf{y}\|\cdot\hat{\mathbf{y}}\ \rangle - \langle\ \Phi(\|\mathbf{x}\|\cdot\hat{\mathbf{x}}), \Phi(\|\mathbf{y}\|\cdot\hat{\mathbf{y}})\ \rangle\big| \\ &= \|\mathbf{x}\|\|\mathbf{y}\|\cdot\big|\langle \mathbf{x},\mathbf{y}\rangle - \langle \Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle\big|, \end{aligned}$$

and after dividing out this product of norms and squaring, we get the result from part (a):

$$|\langle \mathbf{x}_j,\mathbf{x}_k\rangle - \langle \Phi(\mathbf{x}_j),\Phi(\mathbf{x}_k)\rangle| \le \varepsilon\|\mathbf{x}_j\|_2\|\mathbf{x}_k\|_2 \iff |\langle \hat{\mathbf{x}}_j,\hat{\mathbf{x}}_k\rangle - \langle \Phi(\hat{\mathbf{x}}_j),\Phi(\hat{\mathbf{x}}_k)\rangle|^2 \le \|\hat{\mathbf{x}}_j\|_2\|\hat{\mathbf{x}}_k\|_2^2.$$

The matrix Φ precisely satisfies the hypotheses of the Johnson-Lindenstrauss Lemma, which means that

$$(1-\varepsilon)\|\mathbf{x}_j - \mathbf{x}_k\| \le \|\Phi(\mathbf{x}_j) - \Phi(\mathbf{x}_k)\| \le (1+\varepsilon)\|\mathbf{x}_j - \mathbf{x}_k\|$$

occurs with probability at least $1-\delta$ whenever $r \ge c\cdot\varepsilon^{-2}\log(n/\delta)$. This is one of our hypotheses in part (a); to get the second one, we can expand our set χ by taking its union with $-\chi$:

$$\chi' = \chi \cup (-1)\cdot\chi = \{\mathbf{x}_1, ..., \mathbf{x}_n, -\mathbf{x}_1, ..., -\mathbf{x}_n\}.$$

At worst, χ' has twice as many elements as χ, so

$$(1-\varepsilon)\|\mathbf{x}_j \pm \mathbf{x}_k\| \le \|\Phi(\mathbf{x}_j) \pm \Phi(\mathbf{x}_k)\| \le (1+\varepsilon)\|\mathbf{x}_j \pm \mathbf{x}_k\|$$

occurs with probability at least $1-\delta$ whenever $r \ge c\cdot\varepsilon^{-2}\log(2n/\delta)$ (notice the coefficient 2 attached to n). Thus, by part (a), we get that

$$|\langle \mathbf{x}_j,\mathbf{x}_k\rangle - \langle \Phi(\mathbf{x}_j),\Phi(\mathbf{x}_k)\rangle| \le \varepsilon\|\mathbf{x}_j\|_2\|\mathbf{x}_k\|_2.$$

holds with probability at least 0.9 whenever

$$r \ge c\dot{\varepsilon}^{-2}\log(2n/0.1) = \frac{c\log(20n)}{\varepsilon^2}.$$

$\square$

EXERCISE 3. Let A be a square $n \times n$ matrix whose rows are orthonormal. Prove that the columns of A are also orthonormal.

Proof: Since the rows of A are orthonormal, we have that

$$[A \cdot A^\top]_{ij} = A_{i*} \cdot (A^T)_{*j} = A_{i*} \cdot A_{j*} = \begin{cases} 1 & i = j \\ 0 & \text{else} \end{cases}$$

where A_{i*} and $(A^\top)_{*j}$ denote the ith row and jth column of A and $A^\top$ respectively. This then implies that $A \cdot A^\top = I_n$. Because the left and right inverse of an invertible square matrix are the same, $A^\top \cdot A = I_n$ as well and by a calculation similar to the one above we get that

$$(A^\top)_{i*} \cdot A_{*j} = A_{*i} \cdot A_{*j} = \begin{cases} 1 & i = j \\ 0 & \text{else} \end{cases}.$$

Hence, the columns of A are also orthonormal. □

EXERCISE 4. Let A be a square invertible matrix with SVD $\sum_i \sigma_i u_i v_i^\top$. Show that the inverse of A is $\sum_i \sigma_i^{-1} v_i u_i^\top$.

Proof: Note first the because A is invertible it has full rank equal to n, and hence has n left and right singular vectors. Multiplying the proposed SVD of A^{-1} with the SVD of A gives us

$$\begin{aligned} \left(\sum_i \sigma_i u_i v_i^\top\right)\left(\sum_j \sigma_j^{-1} v_j u_j^\top\right) &= \sum_i \left(\sigma_i u_i v_i^\top \cdot \sum_j \sigma_j^{-1} v_j u_j^\top\right) \\ &= \sum_i \sum_j \sigma_i \sigma_j^{-1} (u_i v_i^\top v_j u_j^\top) \\ &= \sum_i u_i u_i^\top, \end{aligned}$$

where all sums range from $1, ..., n$ implicitly. I'm told that one can directly show that the entries of $\sum_i u_i u_i^\top$ give the $n \times n$ identity matrix, but here is a slightly less tedious way to see this. By orthogonality of the u_i, we get that

$$\left(\sum_i u_i u_i^\top\right) u_j = u_j$$

for any $1 \leq j \leq n$. Since the $\{u_1, ..., u_n\}$ vectors span are linearly independent, they span $\mathbb{R}^n$ and we can therefore write any other vector $w \in \mathbb{R}^n$ as a linear combination of the u_i. This implies

$$\left(\sum_i u_i u_i^\top\right) w = w.$$

Lemma 3.3 in the text says that two matrices A and B are equal if and only if for every vector $\mathbf{v}$ $A\mathbf{v} = B\mathbf{v}$, and so by this lemma we conclude that $\sum_i u_i u_i^\top = I_n$.

Because the left and right inverses of invertible square matrices are equal and inverses of square matrices are unique, $A^{-1} = \sum_i \sigma_i^{-1} v_i u^\top$. In matrix notation, we would write

$$A^{-1} = V\Sigma^{-1}U^\top.$$

□

EXERCISE 5. Let $\sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$ be an SVD of a rank r matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ where $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r$. Let $\mathbf{A}_k = \mathbb{R}^{m\times n} = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$ be the rank k approximation to $\mathbf{A}$ obtained by truncating its SVD. Express the following quantities in terms of the singular values $\{\sigma_i\}_{i=1}^r$:

(i) $\|\mathbf{A}\|_F^2$

(ii) $\|\mathbf{A}\|_2^2$

(iii) $\|\mathbf{A} - \mathbf{A}_k\|_F^2$

(iv) $\|\mathbf{A} - \mathbf{A}_k\|_2^2$.

Proof:

(i) Recall that the Frobenius norm of $\mathbf{A}$ is simply the standard Euclidean norm of $\mathbf{A}$ when considered to be a vector in $\mathbb{R}^{m\times n}$, i.e.

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}[\mathbf{A}_{ij}]^2.$$

As written, we are adding the squared norms of each row vector in $\mathbf{A}$. The squared norm of the ith row of $\mathbf{A}$ is the ith diagonal element of $\mathbf{A} \cdot \mathbf{A}^\top$, hence

$$\|\mathbf{A}\|_F^2 = \mathrm{Tr}(\mathbf{A} \cdot \mathbf{A}^\top).$$

This formulation of the Frobenius norm makes it clear (after distributing products and using the pairwise orthogonality of the $\mathbf{v}_i$ a few times) that

$$\begin{aligned}
\|\mathbf{A}\|_F^2 &= \mathrm{Tr}\left(\sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \cdot \left(\sum_{i=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^\top\right)^\top\right) \\
&= \mathrm{Tr}\left(\sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \cdot \sum_{i=1}^{r} \sigma_j \mathbf{v}_j \mathbf{u}_j^\top \cdot\right) \\
&= \mathrm{Tr}\left(\sum_{i=1}^{r} \sigma_i^2 e_i\right) \\
&= \sigma_1^2 + ... + \sigma_r^2
\end{aligned}$$

(ii) The spectral norm of $\mathbf{A}$ is defined

$$\|\mathbf{A}\|_2^2 = \max_{\substack{v\in\mathbb{R}^n \\ \|v\|=1}} |Av|.$$

The right singular vector $\mathbf{v}_1$ of $\mathbf{A}$ is, by definition, the norm 1 vector which maximizes $|Av|$, and σ_1 is this value. Hence,

$$\|\mathbf{A}\|_2^2 = \sigma_1.$$

(iii) The matrix $\mathbf{A} - \mathbf{A}_k$ is a matrix whose SVD is given by $\sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$. Hence, by part *(i)*,

$$\|\mathbf{A} - \mathbf{A}_k\|_F^2 = \sigma_{k+1}^2 + ... + \sigma_r^2.$$

(iv) Similarly, by part *(ii)*,

$$\|\mathbf{A} - \mathbf{A}_k\|_2^2 = \sigma_{k+1}$$

because $\mathbf{A} - \mathbf{A}_k$ is a matrix whose SVD is given by $\sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$.

□

Exercise 6.

(a) Write a program to implement the power method for computing the first right singular vector of a matrix. Apply your program to the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 & \dots & 9 & 10 \\ 2 & 3 & 4 & \dots & 10 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 9 & 10 & 0 & \dots & 0 & 0 \\ 10 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}.$$

(b) Implement a modification of the power method that computes the first four right singular vectors of a matrix, as follows. Randomly select four vectors and find an orthonormal basis for the space spanned by the four vectors. Then multiply each of the basis vectors by $\mathbf{A}\mathbf{A}^\top$ and find a new orthonormal basis for the space spanned by the resulting four vectors. Apply your code to compute the first four singular vectors of the matrix from part (a). Comment on how you might extend this to compute the first k singular vectors of a matrix.

Solution:

(a) Numerical precision poses a problem here, and I actually had better results after implementing this in Rust as it forces you to specify types. Python matched the precision once I told numpy to use float64. The attached python program estimated that the first left singular vector of A was

$$\mathbf{u}_1 = \begin{bmatrix} -0.3197506 \\ -0.36962502 \\ -0.39811309 \\ -0.4039189 \\ -0.38728043 \\ -0.3499587 \\ -0.29512626 \\ -0.22716238 \\ -0.15136864 \\ -0.07362363 \end{bmatrix}$$

by taking the first column of $(A \cdot A^\top)^{16}$ and normalizing. This had an error of $8.007677 * 10^{-9}$ when compared with the first left singular vector given by the numpy.linalg.svd function. The sign of the elements in the estimate $\hat{\mathbf{u}_1}$ for $\mathbf{u}_1$ is manually set to match that of the singular vector calculated by numpy, to ensure the error is accurate. Note that because A is symmetric, the ith left and right singular vectors are the same.

(b) The relevant code for this portion is contained in the function problem6b(). We first generate four random vectors from a standard Gaussian distribution on $\mathbb{R}^{10}$. We then perform QR-factorization to find an orthonormal basis for the subspace spanned by these vectors and calculate the product

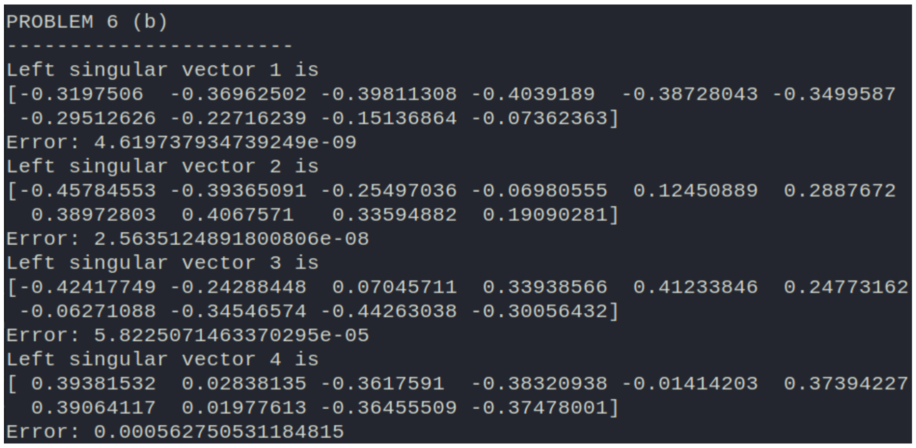
$$A \cdot A^\top \cdot Q$$

from right to left. This is equivalent to calculating the individual matrix-vector products for the column vectors of Q individually. To get a better estimate, we iterate this process: perform QR-factorization on $A \cdot A^\top \cdot Q$ to obtain Q_1 and then find $A \cdot A^T \cdot Q_1$. We did this 16 times, in order to match the iterations in part (a). Finally, we compare the result to numpy.linalg.svd and examine the error. I won't print the estimates for the first four singular vectors of A here; instead, I'll list the errors:

$$\begin{aligned} \mathbf{u}_1 &: \text{ error on the order of } 10^{-9} \\ \mathbf{u}_2 &: \text{ error on the order of } 10^{-8} \\ \mathbf{u}_3 &: \text{ error on the order of } 10^{-5} \\ \mathbf{u}_4 &: \text{ error on the order of } 10^{-4}. \end{aligned}$$

It is once again likely that the error grows as the singular vectors become less significant due to some numerical precision errors.

We could generalize this process to find the first k left singular vectors by first generating k random vectors rather than 4. It is important that we reorthonormalize each time we apply $A \cdot A^\top$ to Q in order to ensure the first singular vector doesn't begin to dominate. Here is an image of my code output for this portion of the problem:



```
PROBLEM 6 (b)
-----------------------
Left singular vector 1 is
[-0.3197506  -0.36962502 -0.39811308 -0.4039189  -0.38728043 -0.3499587
 -0.29512626 -0.22716239 -0.15136864 -0.07362363]
Error: 4.619737934739249e-09
Left singular vector 2 is
[-0.45784553 -0.39365091 -0.25497036 -0.06980555  0.12450889  0.2887672
  0.38972803  0.4067571   0.33594882  0.19090281]
Error: 2.5635124891800806e-08
Left singular vector 3 is
[-0.42417749 -0.24288448  0.07045711  0.33938566  0.41233846  0.24773162
 -0.06271088 -0.34546574 -0.44263038 -0.30056432]
Error: 5.8225071463370295e-05
Left singular vector 4 is
[ 0.39381532  0.02838135 -0.3617591  -0.38320938 -0.01414203  0.37394227
  0.39064117  0.01977613 -0.36455509 -0.37478001]
Error: 0.000562750531184815
```

□

Exercise 7. Read in a grayscale image of your choice, or resolution at least 256 by 256. Perform a singular value decomposition of the matrix. Reconstruct the image using only the 1, 4, 16 and 32 singular values/singular vectors.

(a) Plot the original image along with the reconstructed images.

(b) What percentage of the Frobenius norm is captured in each case?

(c) Repeat steps (a) and (b) above, but now for a "white noise" image of the same dimensions generated as follows: each pixel is generated as an independent uniform random variable on $[0, 1]$. Compare the behavior of the singular values for the noise image and the real image. Please discuss.

Solution:

(a) I plotted this for two images, one a 600×600 image of a rock and another a 900×900 painting of a waterfall (it looks a lot like Yosemite falls to me). Here are the results.

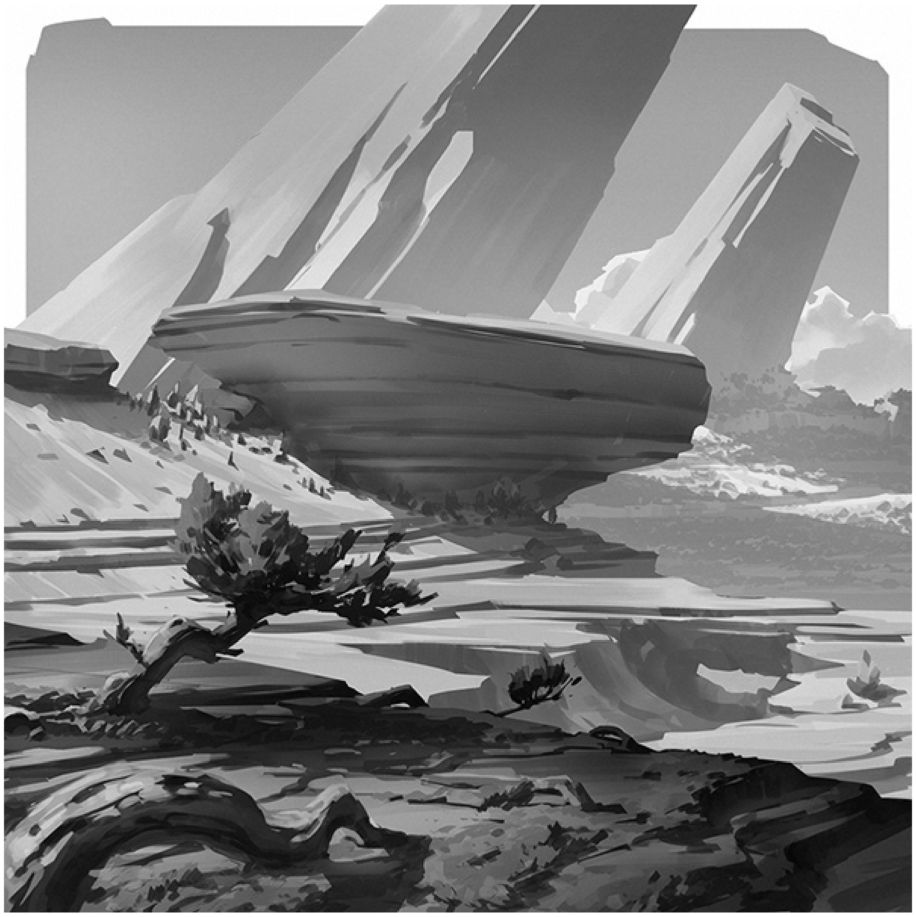


Figure 3: Original image

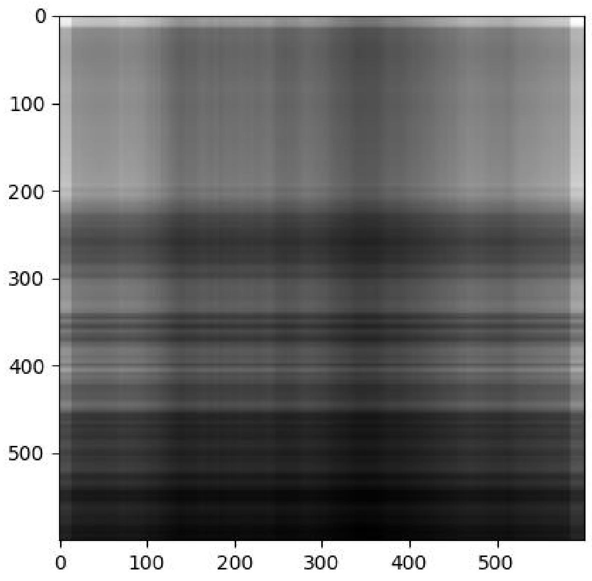


Figure 4: Image reconstructed with 1 singular value, captured 95.258% of the Frobenius norm

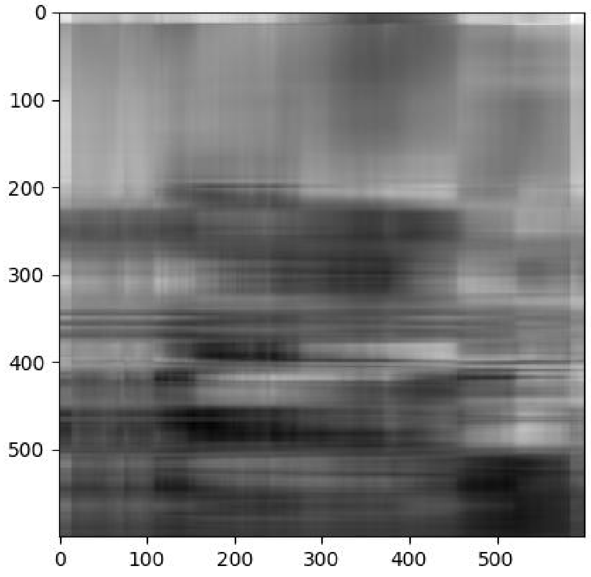


Figure 5: Image reconstructed with 4 singular values, captured 97.421% of the Frobenius norm

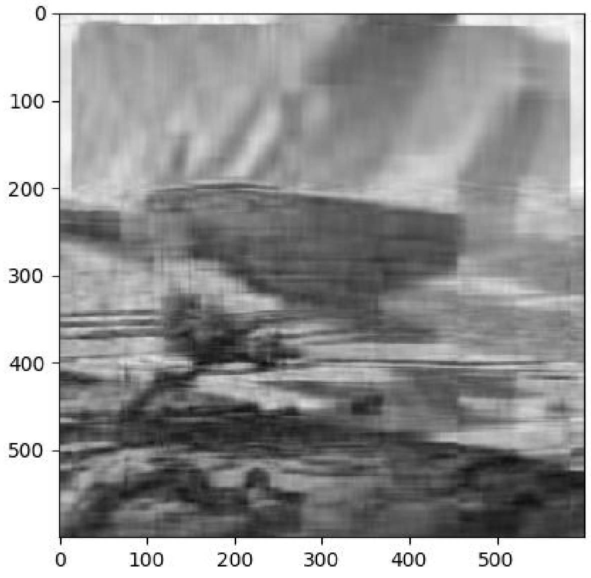


Figure 6: Image reconstructed with 16 singular values, captured 99.060% of the Frobenius norm

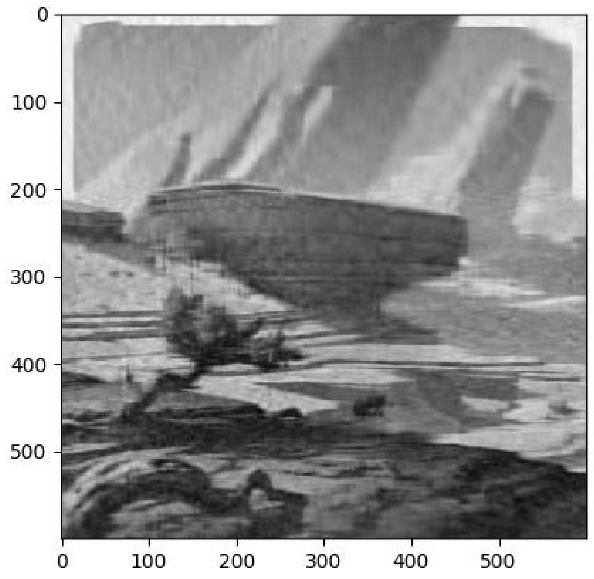


Figure 7: Image reconstructed with 32 singular values, captured 99.464% of the Frobenius norm

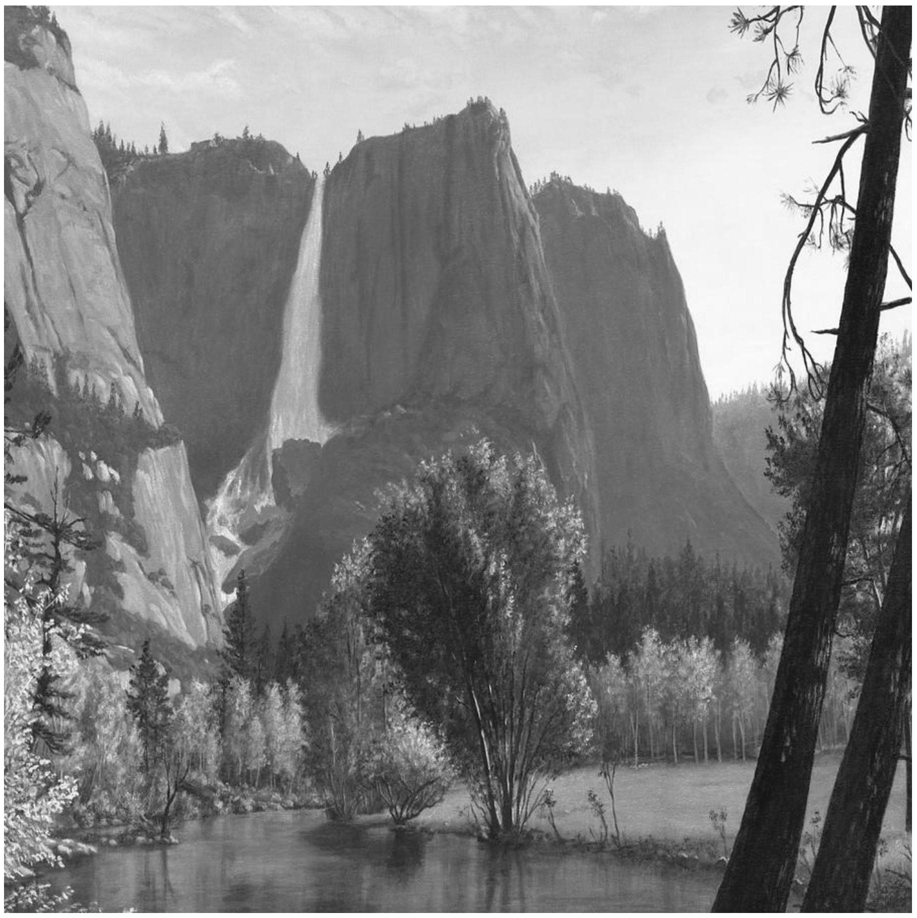


Figure 8: Original image

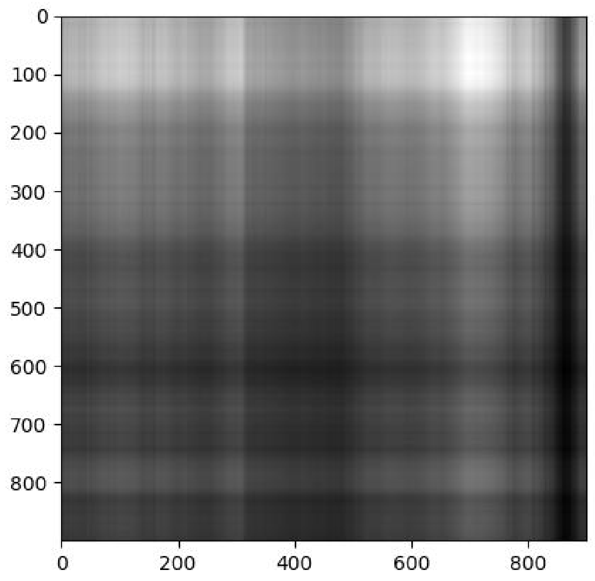


Figure 9: Image reconstructed with 1 singular value, captured 95.345% of the Frobenius norm

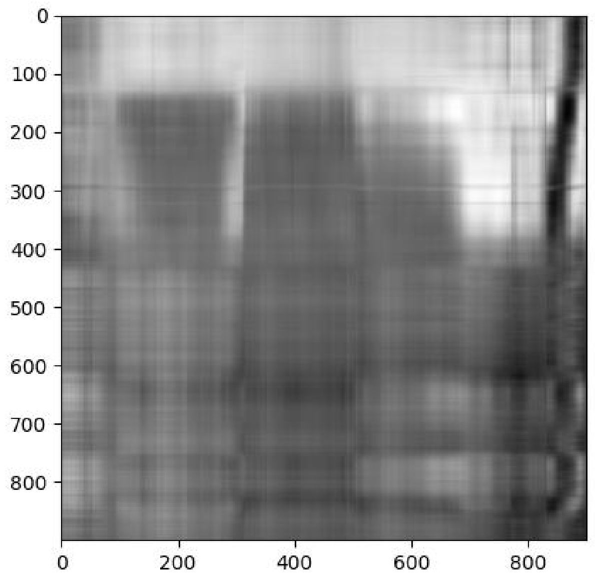


Figure 10: Image reconstructed with 4 singular values, captured 97.835% of the Frobenius norm

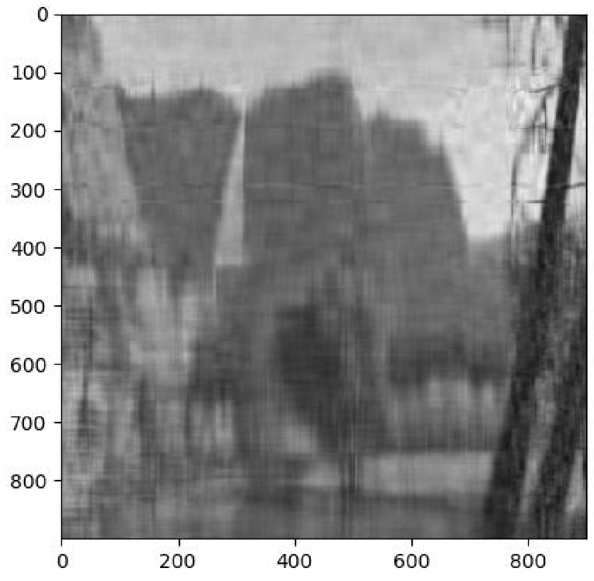


Figure 11: Image reconstructed with 16 singular values, captured 99.012% of the Frobenius norm

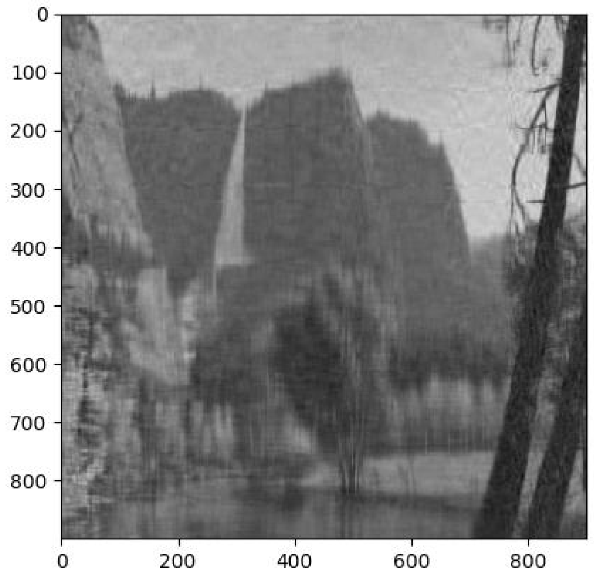


Figure 12: Image reconstructed with 32 singular values, captured 99.351% of the Frobenius norm

(b) The percentage of the Frobenius norm captured in each case is listed in the image captions. However, within a small margin of error, in both cases the percentage of the Frobenius norm captured is 95.3% for 1 singular value, 97.6% for 4 singular values, 99.04% for 16 singular values and 99.40% for 32 singular values. At the very least, this illustrates that closeness in Frobenius norm is a poor measurement of human image recognition. I can't begin to discern details until at least 16 singular values have been incorporated, at which point the images are already 99% similar.

(c) Here are the plots generated for a 600×600 image of random noise.

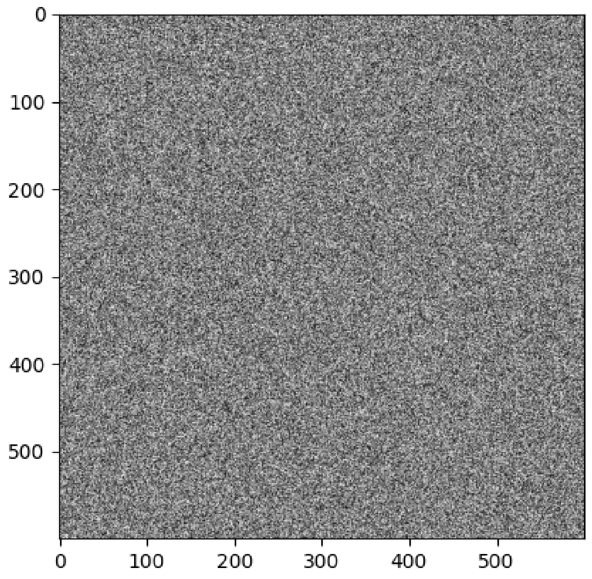


Figure 13: Original image

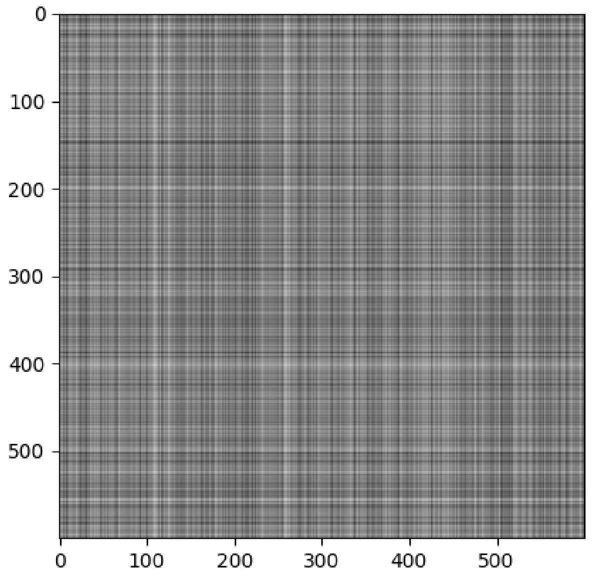


Figure 14: Image reconstructed with 1 singular value, captured 86.646% of the Frobenius norm

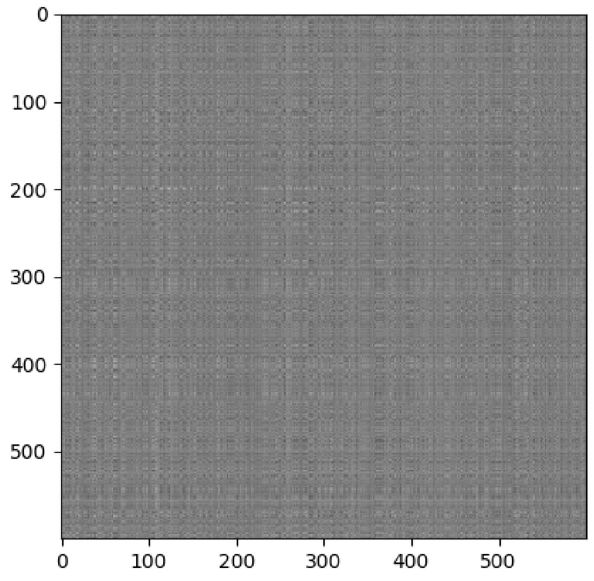


Figure 15: Image reconstructed with 4 singular values, captured 86.925% of the Frobenius norm

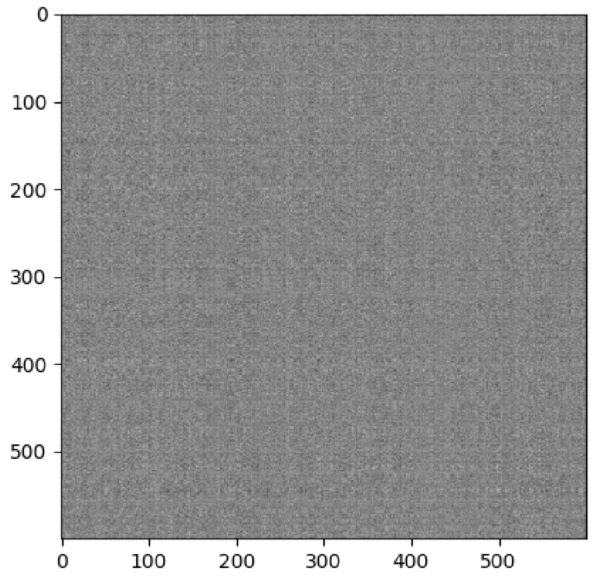


Figure 16: Image reconstructed with 16 singular values, captured 87.945% of the Frobenius norm

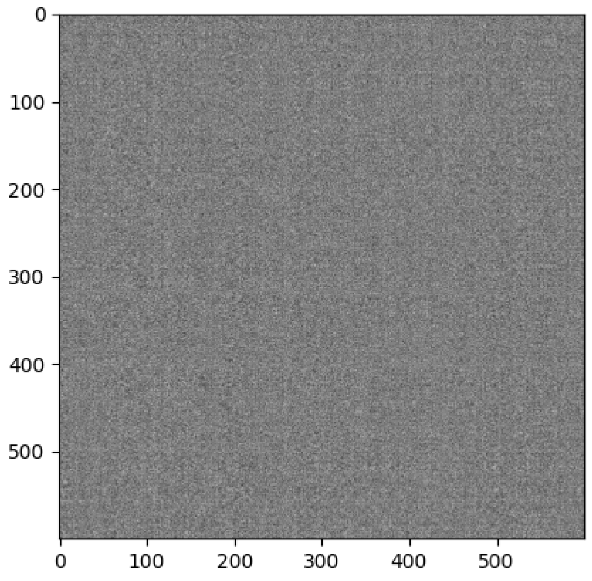


Figure 17: Image reconstructed with 32 singular values, captured 89.155% of the Frobenius norm

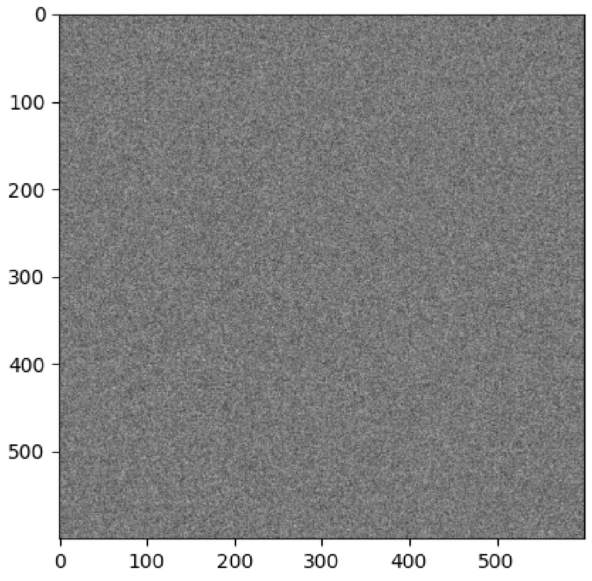


Figure 18: Image reconstructed with 128 singular values, captured 94.324% of the Frobenius norm

The results show that it is far harder for SVD to compress a random image than a "natural" image. Human constructed images and photographs are more likely to have continuously varying color/brightness, i.e. the color of nearby neighboring pixels is likely to match within some margin of error. Loosely, this should mean that regional behavior of the image should capture something about the individual values of a pixel within that region. The image built from random noise features no such correlation between neighboring pixels. To approximate the behavior of an individual pixel, it is therefore necessary to read the value of the pixel itself.

$\square$