Pr. 1. (sol/hsj62)
If $\boldsymbol{A}$ or $\boldsymbol{B}$ is $\boldsymbol{0}$ then the problem is trivial, so assume they are both nonzero.
Denote the compact SVDs of $\boldsymbol{A}$ and $\boldsymbol{B}$ by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$ and $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{\Omega}\boldsymbol{Y}'$, where we omit the usual compact SVD subscripts for simplicity. Here $\boldsymbol{\Sigma}$ and $\boldsymbol{\Omega}$ are square and symmetric and invertible (but possibly different sizes if $\boldsymbol{A}$ and $\boldsymbol{B}$ have different ranks).
Now $\boldsymbol{A}\boldsymbol{B}' = \boldsymbol{0}$ means $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'\boldsymbol{Y}\boldsymbol{\Omega}\boldsymbol{X}' = \boldsymbol{0}$. Multiplying on the left by $\boldsymbol{\Sigma}^{-1}\boldsymbol{U}'$ and on the right by $\boldsymbol{X}\boldsymbol{\Omega}^{-1}$ shows that $\boldsymbol{V}'\boldsymbol{Y} = \boldsymbol{0}$. So $\begin{bmatrix}\boldsymbol{V} & \boldsymbol{Y}\end{bmatrix}$ is a matrix with orthonormal columns.
Likewise $\boldsymbol{A}'\boldsymbol{B} = \boldsymbol{0}$ leads to $\boldsymbol{U}'\boldsymbol{X} = \boldsymbol{0}$, so $\begin{bmatrix}\boldsymbol{U} & \boldsymbol{X}\end{bmatrix}$ is a matrix with orthonormal columns.
Therefore, the following decomposition is a valid compact SVD of $\boldsymbol{A}+\boldsymbol{B}$, to within permutations for sorting the singular values:

$$\boldsymbol{A}+\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}' + \boldsymbol{X}\boldsymbol{\Omega}\boldsymbol{Y}' = \begin{bmatrix}\boldsymbol{U} & \boldsymbol{X}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Omega}\end{bmatrix}\begin{bmatrix}\boldsymbol{V} & \boldsymbol{Y}\end{bmatrix}'.$$

(This "SVD of a sum" can be useful itself.) Thus

$$\|\boldsymbol{A}+\boldsymbol{B}\|_* = \|\boldsymbol{\Sigma}\|_* + \|\boldsymbol{\Omega}\|_* = \|\boldsymbol{A}\|_* + \|\boldsymbol{B}\|_*.$$

Now there is one more subtle point we must address here. When $\boldsymbol{A}$ and $\boldsymbol{B}$ are both $M \times N$, we need to be sure that $\text{size}(\boldsymbol{\Sigma}) + \text{size}(\boldsymbol{\Omega}) \leq \min(M,N)$. Specifically, if $\boldsymbol{U}$ is $M \times r_1$ and $\boldsymbol{X}$ is $M \times r_2$, then we need $r_1 + r_2 \leq M$ for the above compact SVD if $\boldsymbol{A}+\boldsymbol{B}$ to be valid. This inequality is assured by the condition $\boldsymbol{U}'\boldsymbol{X} = \boldsymbol{0}$ because $\boldsymbol{U}$ and $\boldsymbol{X}$ are each orthonormal bases in $\mathbb{F}^M$, so if the sum of their dimensions were to exceed M then their spans would have a nontrivial intersection which would contradict $\boldsymbol{U}'\boldsymbol{X} = \boldsymbol{0}$. Likewise for $\boldsymbol{V}$ and $\boldsymbol{Y}$.

Pr. 2. (sol/hsj31)

(a) $\{\boldsymbol{e}_1, \boldsymbol{e}_4\}$ is an orthonormal basis for the null space of $\boldsymbol{X}$.

(b) $\{\boldsymbol{e}_2, \boldsymbol{e}_3\}$ is an orthonormal basis for the orthogonal complement of the null space of $\boldsymbol{X}$.

(c) The projection of $\boldsymbol{x}$ onto the orthogonal complement of the null space of $\boldsymbol{X}$ is $\boldsymbol{P}^{\perp}_{\mathcal{N}(\boldsymbol{X})}\boldsymbol{x} = \begin{bmatrix}0\\2\\3\\0\end{bmatrix}$

(d) Using: `Y = ones(3,3); (U,s,V) = svd(Y); V0=V[:,2:end]` an orthonormal basis for the null space of $\boldsymbol{Y}$ is $\left\{ \begin{bmatrix}0\\-1\\1\end{bmatrix}/\sqrt{2}, \begin{bmatrix}2\\-1\\-1\end{bmatrix}/\sqrt{6} \right\}$

An orthonormal basis for the orthogonal complement of the null space of $\boldsymbol{Y}$ is $\{\boldsymbol{v}_1\}$, $\boldsymbol{v}_1 = \begin{bmatrix}1\\1\\1\end{bmatrix}/\sqrt{3}$

The projection of $\boldsymbol{w}$ onto the orthogonal complement of the null space of $\boldsymbol{Y}$ is $\boldsymbol{P}^{\perp}_{\mathcal{N}(\boldsymbol{Y})}\boldsymbol{w} = \boldsymbol{v}_1(\boldsymbol{v}_1'\boldsymbol{w}) = \begin{bmatrix}2\\2\\2\end{bmatrix}$

(e) A possible `Julia` implementation is

```
using LinearAlgebra

function orthcompnull(A, x)
#
# Syntax:       z = orthcompnull(A, x)
#
# Inputs:       A : m x n matrix
#               x : vector of length n, or matrix with n rows and many columns
#
# Outputs:      z : vector or matrix of size ??? (you determine this)
#
# Description:  Projects x onto the orthogonal complement of the null space
#               of the input matrix A
#
# For full credit, your solution should be computationally efficient.
```

```
# Parse inputs
x = vec(x) # make sure it is a column

(_, s, V) = svd(A)
r = rank(A) # unfortunate redundancy with svdvals, could avoid by using s

# orthonormal basis for orthogonal complement of the null space of A:
Vr = V[:,1:r]

return Vr * (Vr' * x) # output size is n by the number of columns of x

end
```

The above solution uses a `svd` call. The SVD itself calls a **QR decomposition** that is related to **Gram-Schmidt orthogonalization.** It is likely that an even more efficient solution exists by using the QR decomposition directly to find the basis $\boldsymbol{V}_r$ for the orthogonal complement of the null space of $\boldsymbol{A}$. A QR approach is not required for full credit because we have not covered the QR decomposition.
Graders: accept solutions that use QR instead of SVD, as long as no `inv` or `pinv` or other expensive operations are used and as long as the basis matrix $\boldsymbol{V}_r$ is used efficiently with parentheses like this: `Vr * (Vr' * x)`
Because $\mathcal{N}^{\perp}(\boldsymbol{A}) = \mathcal{R}(\boldsymbol{A}')$, having a basis $\boldsymbol{V}_r$ for the range space of $\boldsymbol{A}'$ suffices. The following code also passes.

```
function orthcompnull(A,x)
(Q,~) = qr(A') # QR approach to getting basis for range(A')
return Q * (Q' * x);
end
```

Pr. 3. (sol/hsj43)
Given training data $(\boldsymbol{x}_n, y_n), n = 1, \ldots, N$ consisting of pairs of features $\boldsymbol{x}_n \in \mathbb{R}^M$ and responses $y_n \in \mathbb{R}$, we train a linear "artificial neuron" to minimize the average MSE loss by solving the following optimization problem:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}), \quad L(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} \|y_n - \boldsymbol{w}'\boldsymbol{x}_n\|_2^2 = \frac{1}{N}\|\boldsymbol{y}' - \boldsymbol{w}'\boldsymbol{X}\|_{\mathrm{F}}^2 = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{w}\|_2^2,$$

where $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_N \end{bmatrix} \in \mathbb{R}^{M \times N}$ and $\boldsymbol{y} = (y_1, \ldots, y_N)' \in \mathbb{R}^N$.
Assuming $\boldsymbol{X}$ has full rank, the LS solution is simply

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}\boldsymbol{X}')^{-1}\boldsymbol{X}\boldsymbol{y} = \boldsymbol{K}_x^{-1}\boldsymbol{K}_{yx}'$$

where $\boldsymbol{K}_x = \frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n\boldsymbol{x}_n'$ and $\boldsymbol{K}_{yx} = \frac{1}{N}\sum_{n=1}^N y_n\boldsymbol{x}_n'$.
We need $N \geq M$ for there to be any chance that $\boldsymbol{K}_x$ is invertible. Otherwise we need a pseudo-inverse or some form of regularization.

Pr. 4. (sol/hsj72)

(a) We already know that $\boldsymbol{B}'\boldsymbol{B} \succeq \boldsymbol{0}$ so to show $\boldsymbol{B}'\boldsymbol{B} \succ \boldsymbol{0}$ we simply must show that $\boldsymbol{x} \neq \boldsymbol{0} \Rightarrow \boldsymbol{x}'\boldsymbol{B}'\boldsymbol{B}\boldsymbol{x} \neq 0$. Suppose there is a nonzero $\boldsymbol{x}$ such that $\boldsymbol{x}'\boldsymbol{B}'\boldsymbol{B}\boldsymbol{x} = 0$. Then $\|\boldsymbol{B}\boldsymbol{x}\| = 0$ so $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{0}$, but this would contradict the linear independence of the columns of $\boldsymbol{B}$.

(b) $\boldsymbol{A} \succ \boldsymbol{0}$ means that all eigenvalues are positive, hence nonzero, so the matrix is invertible.

(c) $\boldsymbol{x}'(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{x} = \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}'\boldsymbol{B}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$.

(d) $\boldsymbol{x}'(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{x} = \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}'\boldsymbol{B}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$ because $\boldsymbol{A}$ is positive definite.

(e) Because $\boldsymbol{B}$ has full column rank, $\boldsymbol{B}'\boldsymbol{B} \succeq \boldsymbol{0}$ so $\boldsymbol{A}'\boldsymbol{A} + \boldsymbol{B}'\boldsymbol{B}$ is positive definite by the previous property and thus invertible by an earlier subproblem.

(f) It suffices to show that $\boldsymbol{A}'\boldsymbol{A}+\boldsymbol{B}'\boldsymbol{B} \succ \mathbf{0}$ when $\mathcal{N}(\boldsymbol{A}) \cap \mathcal{N}(\boldsymbol{B}) = \{\mathbf{0}\}$. We already have shown that $\boldsymbol{A}'\boldsymbol{A}+\boldsymbol{B}'\boldsymbol{B} \succeq \mathbf{0}$ so we just need to verify that $\boldsymbol{x}'(\boldsymbol{A}'\boldsymbol{A}+\boldsymbol{B}'\boldsymbol{B})\boldsymbol{x} \neq 0$ for any $\boldsymbol{x} \neq \mathbf{0}$.

Suppose the contrary, *i.e.*, that $\boldsymbol{x}'(\boldsymbol{A}'\boldsymbol{A}+\boldsymbol{B}'\boldsymbol{B})\boldsymbol{x} = 0$ for some $\boldsymbol{x} \neq \mathbf{0}$. then it follows that $\|\boldsymbol{A}\boldsymbol{x}\| = 0$ and $\|\boldsymbol{B}\boldsymbol{x}\| = 0$ so $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$ and $\boldsymbol{B}\boldsymbol{x} = \mathbf{0}$. But this means that $\boldsymbol{x}$ is in the null space of both $\boldsymbol{A}$ and $\boldsymbol{B}$ and that contradicts the assumption that $\mathcal{N}(\boldsymbol{A}) \cap \mathcal{N}(\boldsymbol{B}) = \{\mathbf{0}\}$.

(g) $\boldsymbol{B} \succ \mathbf{0}$ implies that $\boldsymbol{B}$ is invertible, so if $\boldsymbol{x}$ is any nonzero vector, then $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x}$ is nonzero, because otherwise $\boldsymbol{B}$ would be singular.

For any nonzero vector $\boldsymbol{y}$, $\boldsymbol{A} \succ \mathbf{0} \Rightarrow \boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} > 0$ by definition of a positive definite matrix.
Thus $\boldsymbol{A} \succ \mathbf{0},\ \boldsymbol{B} \succ \mathbf{0} \Rightarrow \boldsymbol{x}'\boldsymbol{B}'\boldsymbol{A}\boldsymbol{B}\boldsymbol{x} > 0,\ \forall \boldsymbol{x} \neq \mathbf{0} \Rightarrow \boldsymbol{B}\boldsymbol{A}\boldsymbol{B} \succ \mathbf{0}$.

Pr. 5. (sol/hsj61)
As described in the course notes, because $\|\cdot\|_2$ and $\|\cdot\|_*$ are untarily invariant:

$$\hat{\boldsymbol{X}} = \boldsymbol{U}_r \hat{\boldsymbol{\Sigma}}_r \boldsymbol{V}_r', \quad \hat{\boldsymbol{\Sigma}}_r = \underset{\boldsymbol{S} \succeq \mathbf{0}}{\arg\min} \frac{1}{2} \|\boldsymbol{\Sigma}_r - \boldsymbol{S}\|_2^2 + \beta \|\boldsymbol{S}\|_*, \quad \boldsymbol{S} = \operatorname{diag}(s_1, \ldots, s_r)$$

So here we must solve

$$\underset{s_1,\ldots,s_r \geq 0}{\arg\min} \left\{ \left(\frac{1}{2} \max_k |\sigma_k - s_k|^2 \right) + \beta \sum_{k=1}^{r} s_k \right\}.$$

Having $s_k > \sigma_k$ would only increase the cost, so we must solve

$$\underset{s_1,\ldots,s_r}{\arg\min} \left\{ \left(\frac{1}{2} \max_k (\sigma_k - s_k)^2 \right) + \beta \sum_{k=1}^{r} s_k \right\}, \text{ s.t. } 0 \leq s_k \leq \sigma_k, \forall k.$$

Consider first the case where $r = 1$, then we have simply

$$\hat{\sigma}_1 = \underset{0 \leq s_1 \leq \sigma_1}{\arg\min} f(s_1, \sigma_1, \beta) = \max(\sigma_1 - \beta, 0), \qquad f(s_1, \sigma_1, \beta) \triangleq \frac{1}{2} (\sigma_1 - s_1)^2 + \beta s_1$$

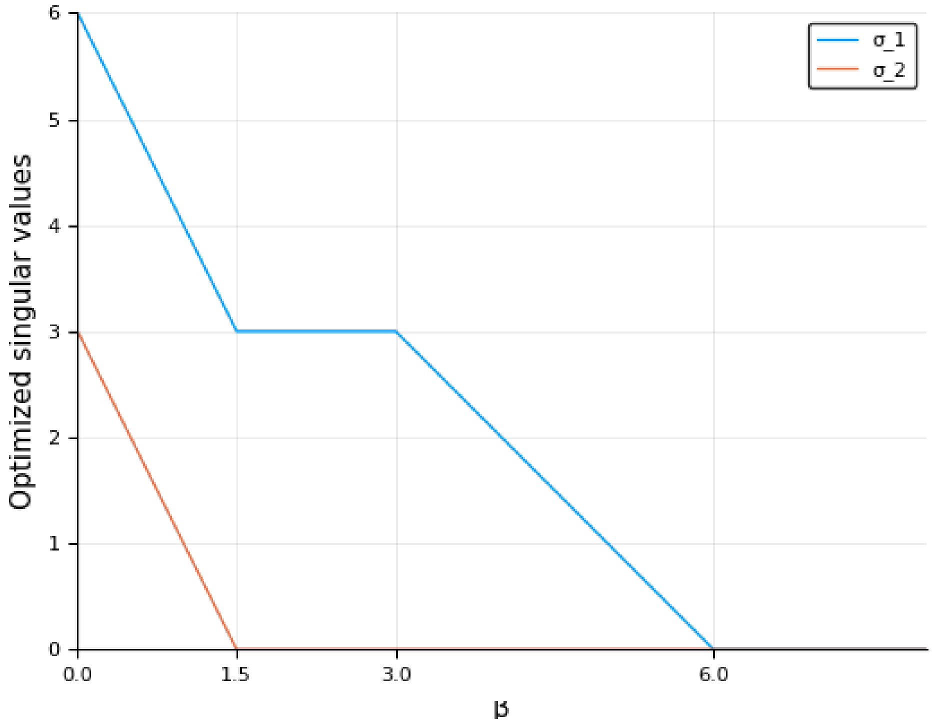
by differentiating and setting to zero and minding the constraints. So the rank-1 minimizer here is

$$\hat{\boldsymbol{X}} = \max(\sigma_1 - \beta, 0) \boldsymbol{u}_1 \boldsymbol{v}_1',$$

which is the same solution as when using the Frobenius norm.
Grader: give full credit for correct solutions to the rank-1 case.
However, for $r = 2$, the solution (found numerically) is different because of the $\max_k$ term:

$$\hat{\sigma}_2 = [\sigma_2 - 2\beta]_+$$
$$\hat{\sigma}_1 = \begin{cases} [\sigma_1 - 2\beta]_+, & 0 \leq \beta \leq \sigma_2/2 \\ \sigma_1 - \sigma_2, & \sigma_2/2 \leq \beta \leq \sigma_2 \\ [\sigma_1 - \beta]_+, & \sigma_2 \leq \beta \leq \sigma_1 \\ 0, & \sigma_1 \leq \beta \end{cases}$$

The following figure illustrates.



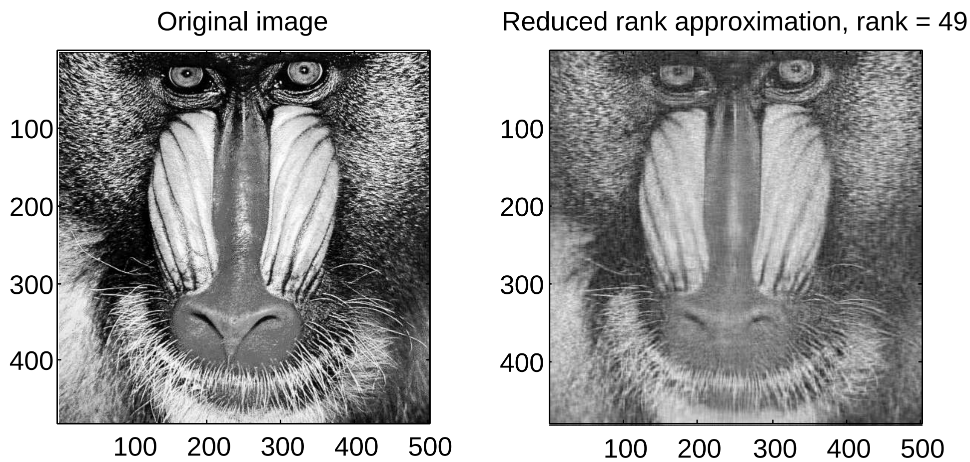
I am unsure how to solve it for a general rank. If you think you have a solution, first have a friend in the class check it, and if you both agree it looks correct, then please tell me about it!

Pr. 6. (sol/hs042)

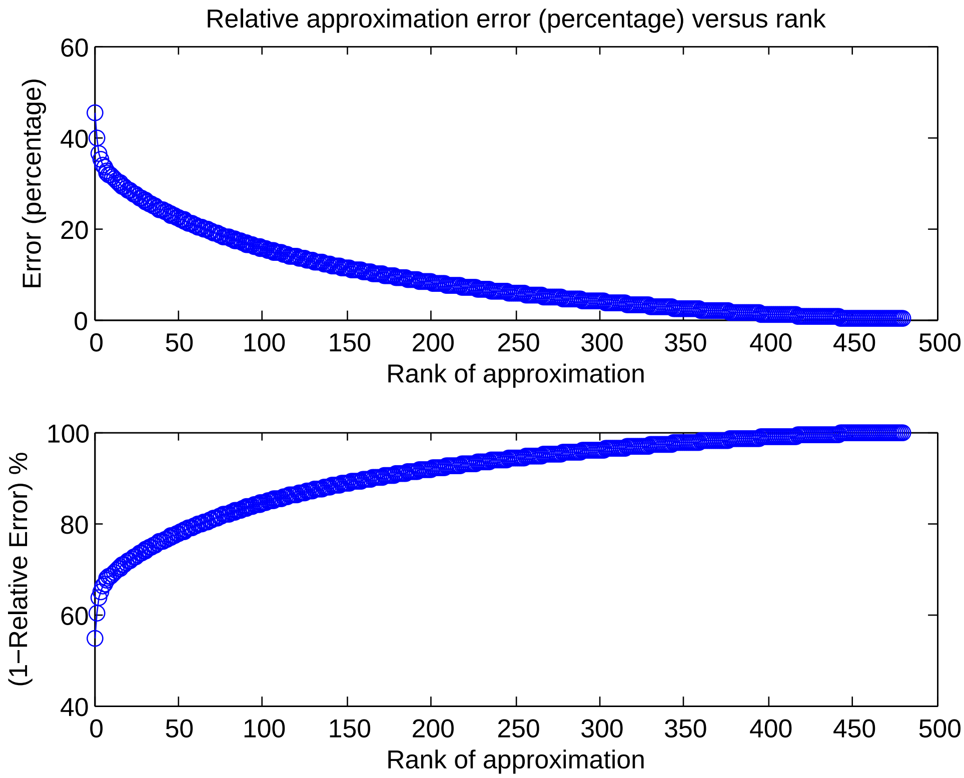
(a) The idea is to view the image as a matrix $\boldsymbol{A} = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i'$ where, for the 500×480 mandrill image shown below, $n = 480$. We know that the optimal (with respect to any unitarily invariant norm) rank-k approximation to $\boldsymbol{A}$ is $\boldsymbol{A}_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i'$. This approximation requires $1+480+500 = 981$ real numbers for each additional term. The original matrix stores $480 \times 500 = 240000$. One-fifth of this total is 48000. Thus the number of terms that gives 48000 real numbers is $48.93 \approx 49$.

Note that if you just took one fifth of the number of terms in the full image, that would give you 96 terms, but that would entail a roughly 40% reduction as opposed to the 20%.

In general, for an $m \times n$ matrix, the rank r approximation requires $r \times (1 + m + n)$ real numbers. Hence, for a given compression fraction p, we want to choose the maximum value of r such that $\frac{r(1+m+n)}{mn} \leq p$. I.e., r can be chosen as the *floor* of $\frac{p \times mn}{1+m+n}$.



(a) true image versus low rank image.



(b) relative error in approximation.

A possible **Julia** implementation is

```
using LinearAlgebra: svd, Diagonal

function compress_image(A, p)
#
# Syntax:       Ac, r = compress_image(A, p)
#
# Inputs:       A is an m x n matrix
#
#               p is a scalar in (0, 1]
#
# Outputs:      Ac is an m x n matrix containing a compressed version of A
#               that can be represented using at most (100 * p)% as many bits
#               required to represent A
#
#               r is the rank of Ac
#
```

```julia
    # Parse inputs
    m, n = size(A)

    # Compute compression factor
    r = Int64(floor(p * m * n / (m + n + 1)))
    r = minimum([r, m, n]) # just to be safe

    # Compute compressed image
    U, s, V = svd(A)
    Ac = U[:, 1:r] * Diagonal(s[1:r]) * V[:, 1:r]'

    return Ac, r
end
```

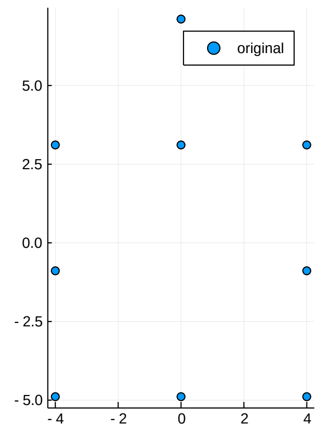
(b) The MIT logo without the lettering has rank 4 if the background white space is represented as a zero in the matrix; otherwise it is rank 5. Note that the singular values σ_k for $k \geq 6$ are not numerically 0, but that they are very, very small. Either way it is nearly perfectly compressible with a low-rank approximation. In contrast, the image of the logo with the lettering is full rank, so the approximation error decreases as the approximation rank increases. Being able to spot low-rank patterns is a valuable skill!

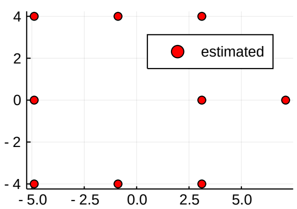
Pr. 7. (sol/hs052)

(a) A possible `Julia` implementation is

```julia
using LinearAlgebra: I, svd, Diagonal

function dist2locs(D, d)
#
# Syntax:       Xr = dist2locs(D, d)
#
# Inputs:       D is an n x n matrix such that D[i, j] is the distance from
#               object i to object j
#
#               d is the desired embedding dimension.
#
# Outputs:      Xr is an n x d matrix whose rows contains the relative
#               coordinates of the n objects
#
# Note:         MDS is only unique up to rotation and translation,
#               so we enforce the following conventions on Xr in this order:
#
#       [ORDER] Xr[:,i] corresponds to ith largest eigenpair of C * C'
#      [CENTER] The centroid of the coordinates is zero
#        [SIGN] The largest magnitude element of Xr[:, i] is positive
#

    # Parse inputs
    n = size(D, 1)

    # Compute correlation matrix - ensures [CENTER]
    S = D .* D
    S = 0.5 * (S + S') # force symmetry (in case of noise)
    P = I - ones(n, n) / n
    CCt = -0.5 * (P * S * P)

    # Compute relative coordinates
    _, s, V = svd(CCt) # using SVD ensures [ORDER]
    Xr = V[:, 1:d] * Diagonal(sqrt.(s[1:d]))

    # Apply [SIGN]
    Xr .*= sign.(Xr[findmax(abs.(Xr), dims=1)[2]])

    return Xr
end
```

(b) Applying the MDS method to the given distance matrix produces the right figure below. The left figure is the original coordinates.





Optional problem(s) below

Pr. 8. (sol/hs034)
Here $\boldsymbol{A} \in \mathbb{R}_8^{19\times 48}$ has rank $r = 8$. We are looking for the number of linearly independent solutions to the homogeneous linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. All vectors $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ must belong to the nullspace of $\boldsymbol{A}$. Thus, the number of linearly independent solutions to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ is precisely the dimension of the nullspace of $\boldsymbol{A}$. From the rank-plus-nullity theorem (Corollary 3.18), we have that $n = \dim \mathcal{N}(A) + \dim \mathcal{R}(A)$. Here $n = 48$, $r = \operatorname{rank}(\boldsymbol{A}) = \dim \mathcal{R}(A) = 8$ so that we must have $\dim \mathcal{N}(A) = 48 - 8 = 40$. Thus there are 40 linearly independent solutions of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$.

Pr. 9. (sol/hs038)
If $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ and $\boldsymbol{B} \in \mathbb{R}^{N\times M}$, then

$$\operatorname{vec}(\boldsymbol{A}^T) = \begin{bmatrix} A(1,:)^T \\ A(2,:)^T \\ \vdots \\ A(M,:)^T \end{bmatrix} \quad \text{and} \quad \operatorname{vec}(\boldsymbol{B}) = \begin{bmatrix} B(:,1) \\ B(:,2) \\ \vdots \\ B(:,M) \end{bmatrix} \Rightarrow \operatorname{vec}(\boldsymbol{A}^T)^T \operatorname{vec}(\boldsymbol{B}) = \sum_{m=1}^{M} A(m,:)B(:,m).$$

Furthermore,

$$\begin{aligned} \operatorname{trace}(\boldsymbol{A}\boldsymbol{B}) &= \operatorname{trace}\left(\begin{bmatrix} A(1,:) \\ A(2,:) \\ \vdots \\ A(M,:) \end{bmatrix} \begin{bmatrix} B(:,1) & B(:,2) & \dots & B(:,M) \end{bmatrix} \right) \\ &= \operatorname{trace}\left(\begin{bmatrix} A(1,:)B(:,1) & A(1,:)B(:,2) & \dots & A(1,:)B(:,M) \\ A(2,:)B(:,1) & A(2,:)B(:,2) & \dots & A(2,:)B(:,M) \\ & & \ddots & \\ A(M,:)B(:,1) & A(M,:)B(:,2) & \dots & A(M,:)B(:,M) \end{bmatrix} \right) \\ &= \sum_{m=1}^{M} A(m,:)B(:,m) = \operatorname{vec}(\boldsymbol{A}^T)^T \operatorname{vec}(\boldsymbol{B}), \end{aligned}$$

using the preceding equality. This is the general property.
When $\boldsymbol{A}$ is symmetric (and $\boldsymbol{B}$ is square with the same size), then $\boldsymbol{A} = \boldsymbol{A}^T$, so that $\operatorname{trace}(\boldsymbol{A}\boldsymbol{B}) = \operatorname{vec}(\boldsymbol{A})^T \operatorname{vec}(\boldsymbol{B})$.

Pr. 10. (sol/hsj51)
Show that the weighted Euclidean norm $\|\boldsymbol{x}\|_{\boldsymbol{W}}$ is a valid norm iff $\boldsymbol{W}$ is a positive definite matrix.
First we show sufficiency: if $\boldsymbol{W}$ is positive definite, then we show that $\|\boldsymbol{x}\|_{\boldsymbol{W}} = \sqrt{\boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}}$ is a valid norm.

- Clearly $\boldsymbol{x} = \boldsymbol{0} \Rightarrow \|\boldsymbol{x}\|_{\boldsymbol{W}} = 0$

- By definition of a positive definite matrix, $\|\boldsymbol{x}\|_{\boldsymbol{W}} = 0 = \sqrt{\boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}} \Rightarrow \boldsymbol{x} = \boldsymbol{0}$
- $\|\alpha\boldsymbol{x}\|_{\boldsymbol{W}} = \sqrt{(\alpha^*)\boldsymbol{x}'\boldsymbol{W}(\alpha\boldsymbol{x})} = |\alpha| \sqrt{\boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}} = |\alpha| \|\boldsymbol{x}\|_{\boldsymbol{W}}$
- Because $\boldsymbol{W}$ is (Hermitian) positive definite, it has a unitary eigendecomposition of the form $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}'$ where the eigenvalues in $\boldsymbol{\Lambda}$ are all positive. Let $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}'$ so that $\boldsymbol{W} = \boldsymbol{S}\boldsymbol{S} = \boldsymbol{S}'\boldsymbol{S}$. Then $\|\boldsymbol{x}\|_{\boldsymbol{W}} = \sqrt{\boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}} = \sqrt{\boldsymbol{x}'\boldsymbol{S}'\boldsymbol{S}\boldsymbol{x}} = \|\boldsymbol{S}\boldsymbol{x}\|$ so $\|\boldsymbol{x}+\boldsymbol{y}\|_{\boldsymbol{W}} = \|\boldsymbol{S}(\boldsymbol{x}+\boldsymbol{y})\| \leq \|\boldsymbol{S}\boldsymbol{x}\| + \|\boldsymbol{S}\boldsymbol{y}\| = \|\boldsymbol{x}\|_{\boldsymbol{W}} + \|\boldsymbol{y}\|_{\boldsymbol{W}}$

Now we show necessity. If $\|\boldsymbol{x}\|_{\boldsymbol{W}}$ is a valid norm, then for all $\boldsymbol{x}$ $0 \leq \|\boldsymbol{x}\|_{\boldsymbol{W}}^2 = \boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}$ and for $\boldsymbol{x} \neq \boldsymbol{0}$: $0 < \|\boldsymbol{x}\|_{\boldsymbol{W}}^2 = \boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}$. These are the two conditions for a (Hermitian) symmetric matrix to be positive definite.
