Pr. 1. (sol/hsj7e)

(a) For a given $\boldsymbol{\pi}$, probably the simplest way (but not the only way) to specify an appropriate transition matrix is

$$\boldsymbol{P}_1 = \boldsymbol{\pi}\mathbf{1}'$$

because for that Markov chain we can easily verify that $\boldsymbol{\pi}$ is indeed an equilibrium distribution:

$$\boldsymbol{P}_1\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{1}'\boldsymbol{\pi} = \boldsymbol{\pi}.$$

A more general option, for which $\boldsymbol{P}_1$ is a special case, is

$$\boldsymbol{P}_\alpha = (1-\alpha)\boldsymbol{I} + \alpha\boldsymbol{\pi}\mathbf{1}', \qquad \alpha \in (0,1].$$

(b) If $\boldsymbol{x}$ is any probability vector such that $\boldsymbol{P}_\alpha\boldsymbol{x} = \boldsymbol{x}$ then $(1-\alpha)\boldsymbol{x} + \alpha\boldsymbol{\pi}\mathbf{1}'\boldsymbol{x} = \boldsymbol{x}$ so $\boldsymbol{x} = \boldsymbol{\pi}$. Thus $\boldsymbol{\pi}$ is the unique equilibrium distribution.

Grader: there might be other $\boldsymbol{P}$ choices where $\boldsymbol{\pi}$ is not the unique equilibrium distribution. If you see any, please let me know!

(c) $\boldsymbol{P}_1\boldsymbol{\pi}_1 = \boldsymbol{\pi}\mathbf{1}'\boldsymbol{\pi}_1 = \boldsymbol{\pi}$, so the power iteration for $\boldsymbol{P}_1$ converges in 1 iteration to $\boldsymbol{\pi}$ for any initial probability vector.
For $\boldsymbol{P}_\alpha$, one can verify that $\boldsymbol{P}_\alpha^k\boldsymbol{\pi}_1 = (1-\alpha)^k(\boldsymbol{\pi}_1 - \boldsymbol{\pi}) + \boldsymbol{\pi}$, which converges to $\boldsymbol{\pi}$ as $k \to \infty$ for $\alpha \in (0,1]$.
Grader: there can be other (more complicated) examples of $\boldsymbol{P}$ for part (a) that could change the answers to (b) and (c). Flag those if you have questions.

Pr. 2. (sol/hsj7g)
By experimenting numerically or by examining the graph, it becomes clear that the equilibrium distribution has the form $\boldsymbol{\pi}_* = (\pi_1, \pi_2, \pi_1, \ldots, \pi_1)$ where $\pi_2 = p\pi_1$, so $\boldsymbol{\pi}_* = (\pi_1, p\pi_1, \pi_1, \ldots, \pi_1)$. The elements of $\boldsymbol{\pi}_*$ must sum to unity, so $(N-1)\pi_1 + p\pi_1 = 1 \Rightarrow \pi_1 = 1/(N-1+p)$. Thus

$$\boldsymbol{\pi}_* = \left(\frac{1}{N-1+p}, \frac{p}{N-1+p}, \frac{1}{N-1+p}, \ldots \frac{1}{N-1+p}\right) = \frac{1}{N-1+p}\mathbf{1}_N + (p-1)\boldsymbol{e}_2.$$

The graph for this Markov chain is strongly connected, so $\boldsymbol{P}$ is irreducible, so $\boldsymbol{\pi}_*$ is unique.
To confirm mathematically that $\boldsymbol{P}\boldsymbol{\pi}_* = \boldsymbol{\pi}_*$, where $\boldsymbol{\pi}_* = ((p-1)\boldsymbol{e}_2 + \mathbf{1})\pi_1$, first note that

$$\boldsymbol{G}_N\boldsymbol{\pi}_* = (\pi_1, \pi_1, \pi_2, \pi_1, \ldots, \pi_1) = ((p-1)\boldsymbol{e}_3 + \mathbf{1})\pi_1.$$

Thus

$$\begin{aligned}\boldsymbol{P}\boldsymbol{\pi}_* &= ((p-1)\boldsymbol{e}_3 + \mathbf{1})\pi_1 + (1-p)(\boldsymbol{e}_3 - \boldsymbol{e}_2)\boldsymbol{e}_1'((p-1)\boldsymbol{e}_2 + \mathbf{1})\pi_1 \\ &= ((p-1)\boldsymbol{e}_3 + \mathbf{1})\pi_1 + (1-p)\pi_1(\boldsymbol{e}_3 - \boldsymbol{e}_2) = \mathbf{1}\pi_1 + (p-1)\pi_1\boldsymbol{e}_2 = (\mathbf{1} + (p-1)\boldsymbol{e}_2)\pi_1 = \boldsymbol{\pi}_*.\end{aligned}$$

Pr. 3. (sol/hsj82)
If $\boldsymbol{x}, \boldsymbol{z} \in \mathcal{B}_r \triangleq \left\{\boldsymbol{x} \in \mathbb{F}^N : \|\boldsymbol{x}\| \le r\right\}$, then $\|\boldsymbol{x}\| \le r$ and $\|\boldsymbol{z}\| \le r$. Thus for any $\alpha \in [0,1]$, using the triangle inequality:

$$\|\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{z}\| \le \|\alpha\boldsymbol{x}\| + \|(1-\alpha)\boldsymbol{z}\| = |\alpha|\,\|\boldsymbol{x}\| + |1-\alpha|\,\|\boldsymbol{z}\| = \alpha\,\|\boldsymbol{x}\| + (1-\alpha)\,\|\boldsymbol{z}\| \le \alpha r + (1-\alpha)r = r,$$

where the 2nd equality used the fact that α is real and between 0 and 1. So $\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{z} \in \mathcal{B}_r$, showing that $\mathcal{B}_r$ is a convex set.

Pr. 4. (sol/hsj81)

(a) $\psi(t) \triangleq (\max(1-t, 0))^2 = \begin{cases}(1-t)^2, & t < 1 \\ 0, & t \ge 1,\end{cases}$ so $\dot{\psi}(t) = \begin{cases}2(t-1), & t < 1 \\ 0, & t \ge 1\end{cases}$ and $\ddot{\psi}(t) = \begin{cases}2, & t < 1 \\ 0, & t > 1 \\ \text{undefined} & t = 1.\end{cases}$

By plotting $\dot{\psi}(t)$, we see that its maximum slope is $L_{\dot{\psi}} = 2$.

(b) Following the course notes, $L = L_{\dot{\psi}}\,\|\boldsymbol{A}\|_2^2 + \beta = 2\,\|\boldsymbol{A}\|_2^2 + \beta$, where $\boldsymbol{A}$ is the $M \times N$ matrix with mth row $y_m\boldsymbol{v}_m^T$.

Saturday December 18 2021 17:36 yuzhang@umich.edu

Pr. 5. (sol/hsj91)

We seek the form $\boldsymbol{A} = \alpha \mathbf{1}_3 \mathbf{1}_4' + \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} d & e & f & g \end{bmatrix}$.

Using the first two columns and the first and third rows, we have $\frac{20-\alpha}{8-\alpha} = \frac{14-\alpha}{6-\alpha}$ so $\alpha = 2$.

So now we have $\boldsymbol{A} - 2\mathbf{1}_3 \mathbf{1}_4' = \begin{bmatrix} 18 & 12 & w-2 & 15 \\ 42 & x-2 & 14 & y-2 \\ 6 & 4 & z-2 & 5 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} d & e & f & g \end{bmatrix}$.

$42/18 = (x-2)/12$ so $x = 30$.

Similar reasoning for the others leads $\boldsymbol{A} - 2\mathbf{1}_3 \mathbf{1}_4' = \begin{bmatrix} 18 & 12 & 6 & 15 \\ 42 & 28 & 14 & 35 \\ 6 & 4 & 4 & 5 \end{bmatrix}$

Thus $\boldsymbol{A} = \begin{bmatrix} 20 & 14 & 8 & 17 \\ 44 & 30 & 16 & 37 \\ 8 & 6 & 4 & 7 \end{bmatrix} = 2\mathbf{1}_3 \mathbf{1}_4' + \begin{bmatrix} 3 \\ 7 \\ 1 \end{bmatrix} \begin{bmatrix} 6 & 4 & 2 & 5 \end{bmatrix}$, with $w = 8,\ x = 30,\ y = 37,\ z = 4$.

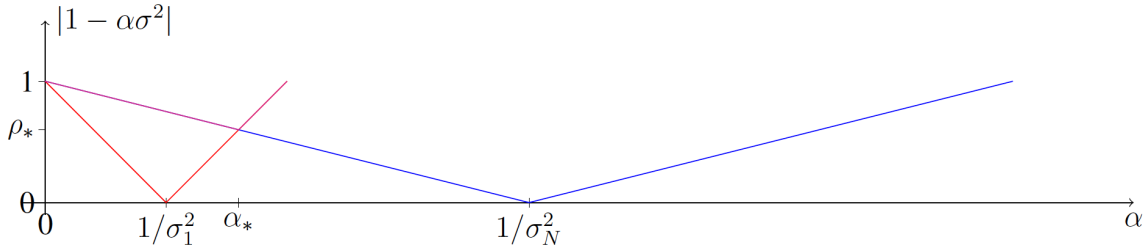
Pr. 6. (sol/hsj71)

Convergence of classical gradient descent is governed by the eigenvalues of $\boldsymbol{G} = \boldsymbol{I} - \alpha \boldsymbol{A}'\boldsymbol{A}$.
Those eigenvalues are $\operatorname{eig}(\boldsymbol{G}) = \operatorname{eig}(\boldsymbol{I} - \alpha \boldsymbol{A}'\boldsymbol{A}) = 1 - \alpha \operatorname{eig}(\boldsymbol{A}'\boldsymbol{A}) = 1 - \alpha\sigma_1^2 \le \ldots \le 1 - \alpha\sigma_N^2$, for $\boldsymbol{A} \in \mathbb{C}^{M \times N}$.
So $\rho(\boldsymbol{G}) = \rho(\boldsymbol{I} - \alpha \boldsymbol{A}'\boldsymbol{A}) = \max(\left|1 - \alpha\sigma_1^2\right|, \left|1 - \alpha\sigma_N^2\right|)$

(a) To minimize $\rho(\boldsymbol{G})$, we need to find $\alpha_* = \arg\min_\alpha \max(\left|1 - \alpha\sigma_1^2\right|, \left|1 - \alpha\sigma_N^2\right|)$. The solution is when $-(1 - \alpha_*\sigma_1^2) = 1 - \alpha_*\sigma_N^2$. Solving for α_* yields:

$$\alpha_* = \frac{2}{\sigma_1^2(\boldsymbol{A}) + \sigma_N^2(\boldsymbol{A})}.$$

The following figure illustrates:



(b) Substituting in the optimal α yields:

$$\rho(\boldsymbol{I} - \alpha_* \boldsymbol{A}'\boldsymbol{A}) = 1 - \alpha_*\sigma_N^2 = 1 - \frac{2}{\sigma_1^2 + \sigma_N^2}\sigma_N^2 = \frac{\sigma_1^2 - \sigma_N^2}{\sigma_1^2 + \sigma_N^2} = \frac{\kappa - 1}{\kappa + 1},$$

where $\kappa \triangleq \sigma_1^2/\sigma_N^2 \ge 1$ denotes the condition number of $\boldsymbol{A}'\boldsymbol{A}$. Better conditioned problems (smaller κ values) lead to faster convergence of GD.

Pr. 7. (sol/hsj79)

(a) Following the analysis of PGD in the notes, the convergence of $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \boldsymbol{P}_0 \boldsymbol{A}'(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$ is governed by the eigenvalues of

$$\boldsymbol{I} - \alpha \boldsymbol{P}_0^{1/2} \boldsymbol{A}'\boldsymbol{A} \boldsymbol{P}_0^{1/2}.$$

Following the analysis in another HW problem, the optimal step size is

$$\alpha_* = \frac{2}{\sigma_1^2(\boldsymbol{A}\boldsymbol{P}_0^{1/2}) + \sigma_N^2(\boldsymbol{A}\boldsymbol{P}_0^{1/2})} = \frac{2}{\sigma_1(\boldsymbol{P}_0^{1/2}\boldsymbol{A}'\boldsymbol{A}\boldsymbol{P}_0^{1/2}) + \sigma_N(\boldsymbol{P}_0^{1/2}\boldsymbol{A}'\boldsymbol{A}\boldsymbol{P}_0^{1/2})}.$$

(b) When $\boldsymbol{P}_0 = (\boldsymbol{A}'\boldsymbol{A})^{-1}$ we have $\boldsymbol{P}_0^{1/2} = (\boldsymbol{A}'\boldsymbol{A})^{-1/2}$ so $\boldsymbol{P}_0^{1/2}\boldsymbol{A}'\boldsymbol{A}\boldsymbol{P}_0^{1/2} = \boldsymbol{I}$ so $\alpha_* = \frac{2}{1+1} = 1$, which makes sense because PGD converges in 1 iteration from any starting point when we use the ideal preconditioner.

Grader: this 2nd part is not graded in F20.

Saturday 18 Dec 2021 17:36 yuzhe@umich.edu

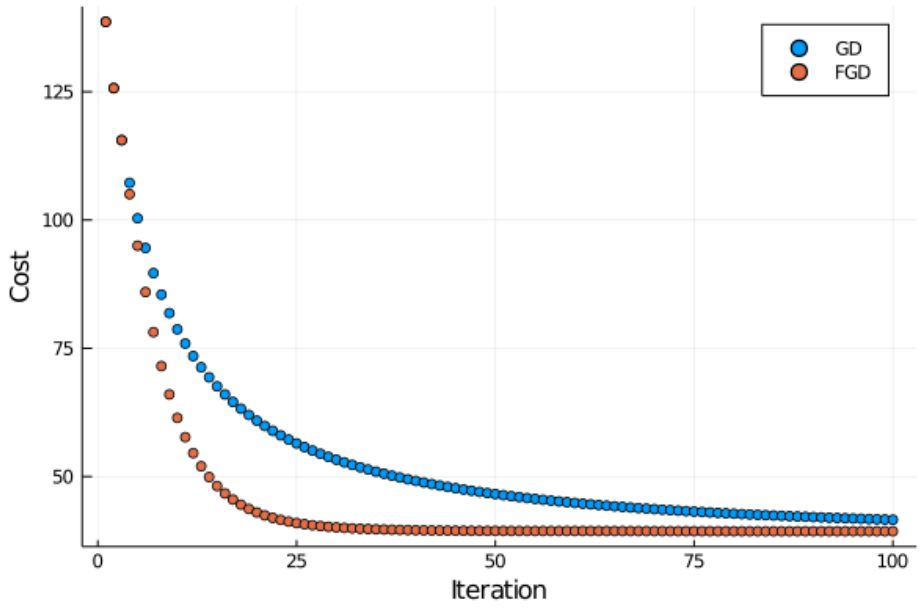
Pr. 8. (sol/hsj05)
Thank you for sharing your reflection.
Graders: assign credit for any submission that looks sincere. (Prof. Fessler will read later.)

Pr. 9. (sol/hsj90)
Thank you for your feedback via the course evaluations.

Saturday December 18 2021 17:36
yuzhanj@umich.edu

Pr. 10. (sol/hsj8t)

(a) Part 1. $L = 23.98$



Grader: they only need a single curve: for the FGD cost function.

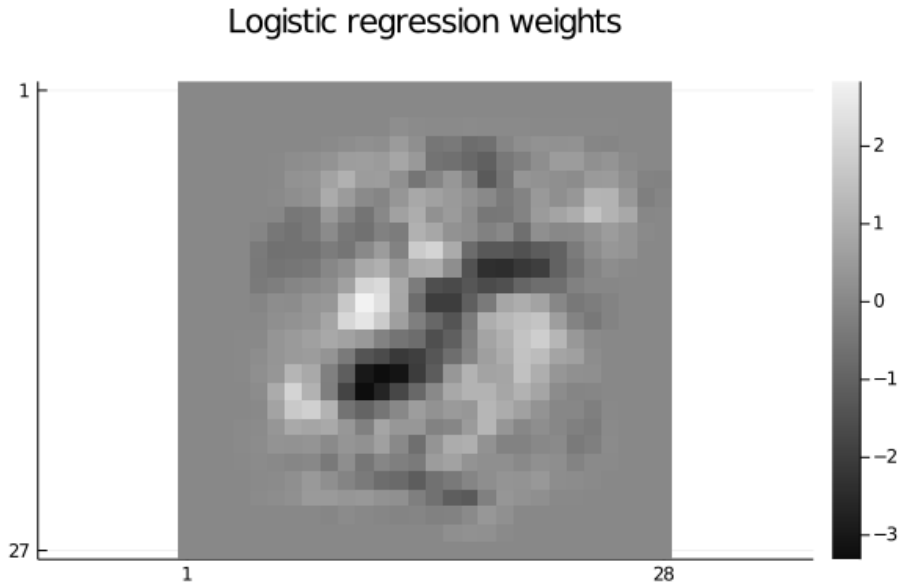
Although a more rigorous way to test convergence is to look at the change in the descent variable, we often judge convergence by where the cost function flattens out. Here, we see that the FGD cost function levels out around iteration 30, so the cost function has converged in 100 iterations. In comparison, gradient descent has not yet converged (is still descending) in 100 iterations.

A possible `Julia` implementation is

```
"""
    x, out = ngd(grad, x0 ; nIters::Int = 200, L::Real = 0, fun = (x, iter) -> 0)

Implementation of Nesterov's FGD (fast gradient descent),
given the gradient of the cost function

In:
- `grad` function that takes `x` and calculates the gradient of the cost function with respect to `x`
- `x0` an initial point

Option:
- `nIters` number of iterations
- `L` Lipschitz constant of the derivative of the cost function
- `fun` a function to evaluate every iteration

Out:
- `x` is the guess of the minimizer after running `nIters` iterations
- `out` is an Array of evaluations of the `fun` function
"""
function ngd(grad::Function, x0::AbstractArray ;
    niter::Int = 200, L::Real = 0, fun::Function = (x, iter) -> 0)

    fun_x0 = fun(x0, 0)
    out = similar(Array{typeof(fun_x0)}, niter+1)
    out[1] = fun_x0

    x = copy(x0)
    xold = copy(x0)
    told = 1
    z = copy(x)

    for iter=1:niter
        t = (1 + sqrt(1 + 4 * told^2)) / 2
        x = z - grad(z) / L
```

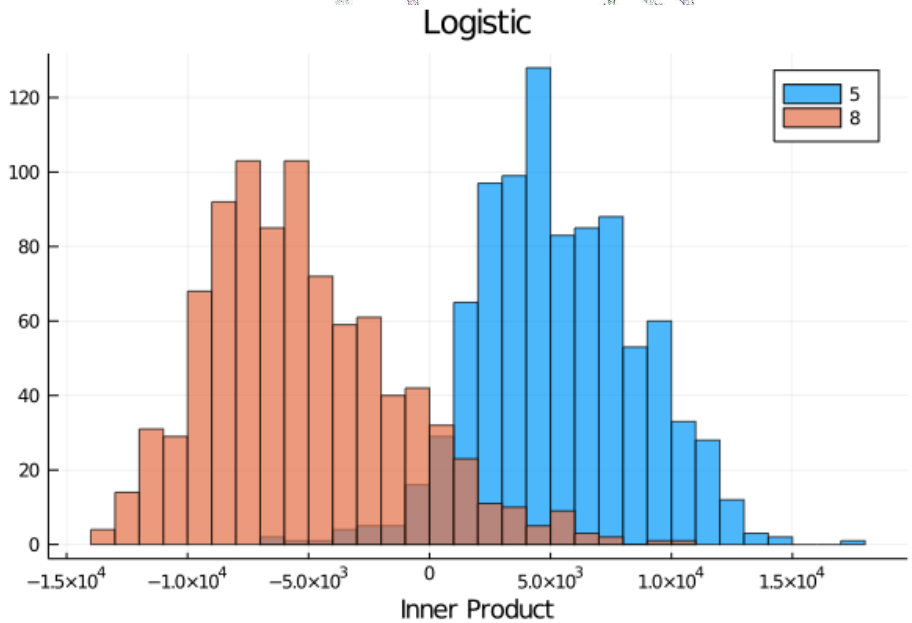
```
        z = x + (told - 1) / t * (x - xold)
        told = copy(t)
        xold = copy(x)
        out[iter+1] = fun(x, iter) # compute cost each iteration
    end

    return x, out
end
```

(b) Part 2



For the logistic regression classifier, the classification accuracy is:
96.22% for the 5 digits
89.22% for the 8 digits

Histograms of inner products:



(c) Part 3

When using 3 basis vectors, the classification accuracy is:

91.11% for the 5 digits
87.44% for the 8 digits

Deciding on how many basis vectors to use is subjective. A common way of deciding is to look for the "knee" in the scree plot, which could be somewhere around 15-30 singular values. Another way would be to look for a large gap in singular values, which would suggest using only 3-7 basis vectors. Ch 6 discusses more formal methods of choosing the number of basis vectors, using permutation or SURE.

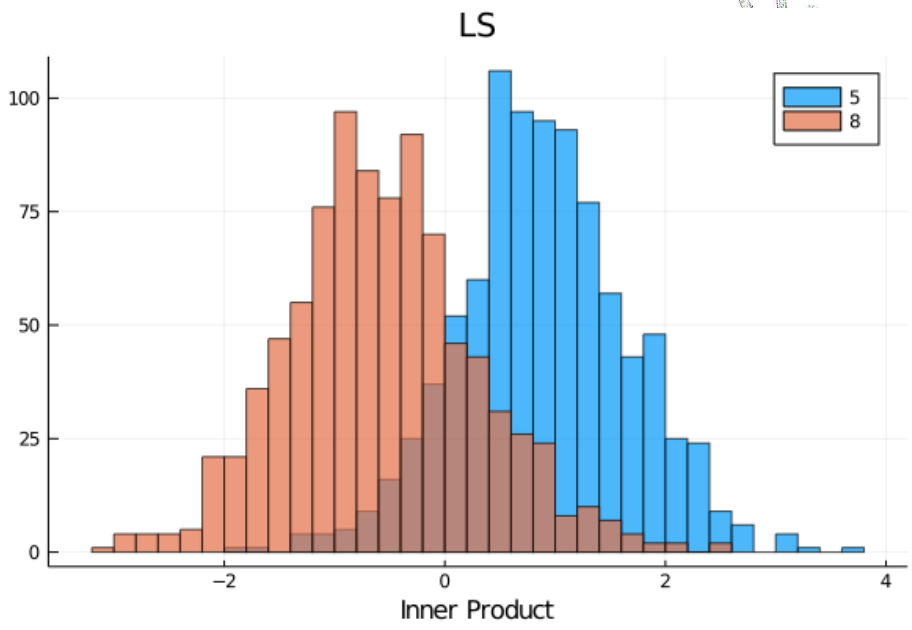
As an example, the classification accuracy when using 20 basis vectors is:
96.22% for the 5 digits
94.44% for the 8 digits

Graders: give credit for any reasonable explanation of how they chose the number of basis vectors to use for the second set of classification accuracies and any corresponding classification accuracies >87%.

(d) Part 4

Classification accuracy is:
88.67% for the 5 digits
77.22% for the 8 digits

Histograms of inner products for LS method:



The least squares based classifier has worse classification accuracy than the logistic regression classifier. This result corresponds to more inner products of test "5" digits with the learned weights being negative and more test "8" digits having a positive inner product.

The block of code in the notebook is classifying the training data. We use 200 total training samples to learn a vector of length 756. Thus, the $\boldsymbol{A}$ matrix in the least-squares problem is wide, the problem is under-determined, and we can perfectly fit the training data. A better cost function would regularize the weights, as discussed in Ch. 4.

(e) Part 5:

Classification accuracy is:
92.89% for the 5 digits
94.44% for the 8 digits

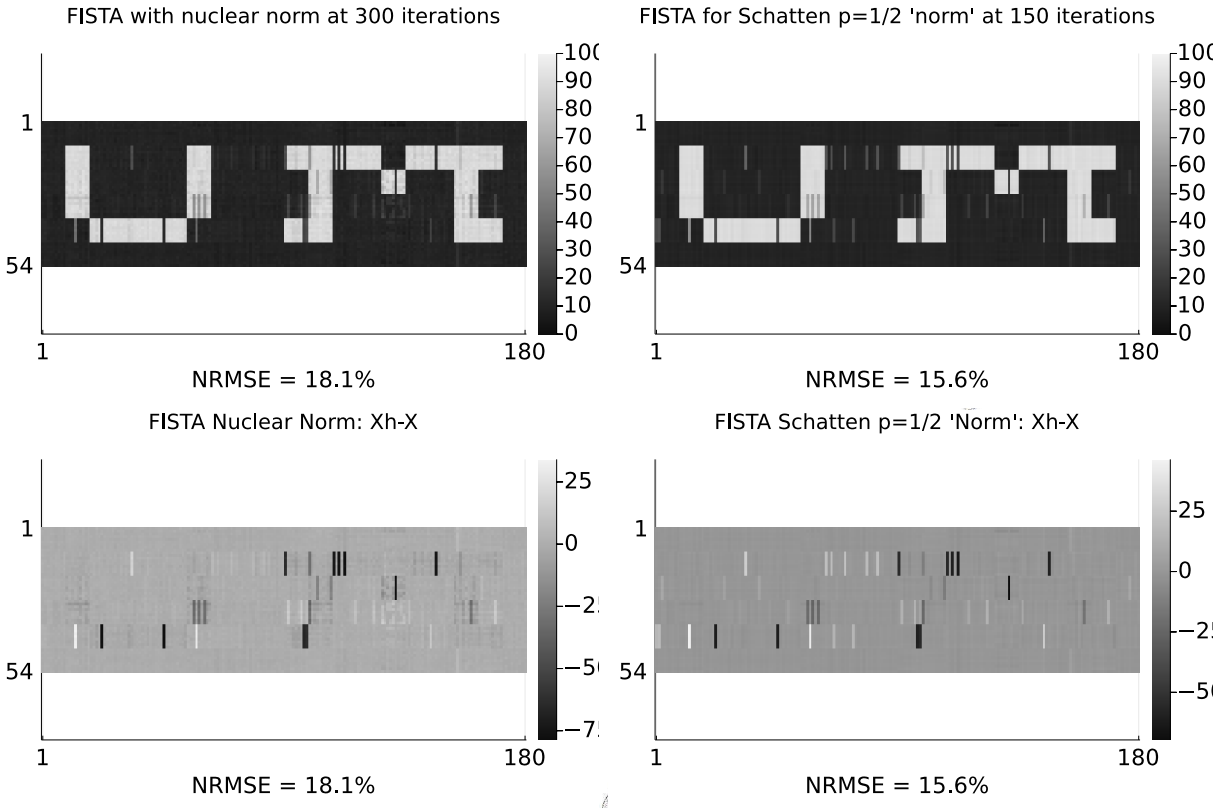
Non-graded problem(s) below

Pr. 11. (sol/hsj69)
The only change to the code needed is to replace SVST (singular value soft thresholding) with the Schatten-based low-rank approximation method.

(a) A possible `Julia` implementation is

```
using LinearAlgebra: svd, Diagonal

"""
    Xh = fista_schatten(Y, M, reg::Real, niter::Int)

Perform matrix completion by using `niter` FISTA iterations
to seek the minimizer over `X` of
`1/2 \\| M .* (Y - X) \\|^2 + reg R(x)`
where `R(X)` is the Schatten p-norm of `X` raised to the `p`th power,
for `p=1/2`, i.e., `R(X) = \\sum_k (\\sigma_k(X))^(1/2)`.

In:
- `Y` : `M × N` matrix (with zeros in missing data locations)
- `M` : `M × N` boolean matrix (`true` in sampled data locations)
- `reg` : regularization parameter
- `niter` : # of iterations

Output
- `Xh` `M × N` estimate of `X` after `niter` FISTA iterations
"""
function fista_schatten(Y, M::AbstractMatrix{Bool}, reg::Real, niter::Int)

    X = copy(Y) # initial guess is just zero-filled data
    Z = copy(X)
    Xold = copy(X)
    told = 1
    for iter=1:niter
        Z[M] = Y[M]
        X = lr_schatten_sol_local(Z, reg) # shrink it
        t = (1 + sqrt(1 + 4*told^2)) / 2
        Z = X + ((told-1)/t) * (X - Xold)
        Xold = X
        told = t
    end

    return X
end


function lr_schatten_sol_local(Y, reg::Real)
    (U,s,V) = svd(Y)
    sh = shrink_p_1_2_sol_local(s, reg)
    Xh = U * Diagonal(sh) * V'
    return Xh
end

function shrink_p_1_2_sol_local(y, reg::Real)
    xh = zeros(size(y))
    fun = (y) -> 4/3 * y * cos(1/3. * acos(- (3^(3/2)*reg) / (4*y^(3/2))))^2
    big = y .> 3/2 * reg^(2/3)
    xh[big] .= fun.(y[big])
    return xh
end
```

(b) The figures below compare the results of FISTA with nuclear norm regularization and with Schatten-based regularization. I did not spend a lot of time tuning the choice of the regularization parameters; it is likely that other parameters could give better results for either method. The recent literature claims that using $p < 1$ can lead to better results than $p = 1$ and this basic experiment seems to concur.



Saturday Decembe
yuzhanj@umi