Homework #2, EECS 598-006, W20. Due **Thu. Jan. 23**, by 4:00PM

---

1. [6] Another **potential function** that is used for edge-preserving regularization is the **hyperbola**:

$$\psi(z) = \delta^2 \left(\sqrt{1 + |z/\delta|^2} - 1 \right).$$

   (a) [3] Determine the **Lipschitz constant** of the derivative of this ψ, *i.e.*, a finite value of L such that

$$\left| \dot{\psi}(s) - \dot{\psi}(t) \right| \leq L \, |s - t| \,, \; \forall s, t \in \mathbb{C}.$$

   (b) [3] Determine the **weighting function** $\omega_\psi(z) = \dot{\psi}(z) \, / z$ of this ψ, for any $z \in \mathbb{C}$.

---

2. [3] Consider the **regularized least-squares** problem

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathbb{R}^N}{\arg\min} \, f(\boldsymbol{x}), \quad f(\boldsymbol{x}) \triangleq \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2 + \beta \frac{1}{2} \left\| \boldsymbol{T}\boldsymbol{x} \right\|_2^2$$

   for $M \times N$ matrix $\boldsymbol{A}$, $K \times N$ matrix $\boldsymbol{T}$, length-M vector $\boldsymbol{y}$, and $\beta \geq 0$, all of which are real.

   For a given $N \times N$ matrix $\boldsymbol{P}$, the **preconditioned steepest descent** algorithm for this problem is:

$$\begin{aligned} \boldsymbol{d}_k &= -\boldsymbol{P} \nabla f(\boldsymbol{x}_k) \\ \alpha_k &= \underset{\alpha \in \mathbb{R}}{\arg\min} \, f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k. \end{aligned}$$

   Find an expression for the step size α_k that is easily implemented. It will depend on $\boldsymbol{x}_k$, $\boldsymbol{d}_k$, $\boldsymbol{A}$, $\boldsymbol{T}$, β and $\boldsymbol{y}$.

   Optional: How does the answer change when $\boldsymbol{x}$,$\boldsymbol{y}$,$\boldsymbol{A}$,$\boldsymbol{T}$ are complex?

---

3. [3] Let $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K$ denote a set of K **unitary** matrices of size $N \times N$, and define the **tight frame** $\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{U}_1 & \ldots & \boldsymbol{U}_K \end{bmatrix}$. We want to represent a vector $\boldsymbol{x} \in \mathbb{F}^N$ as a linear combination of the columns of $\boldsymbol{\Phi}$ using a coefficient vector $\boldsymbol{z}$, *i.e.*, $\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{z}$, where $\boldsymbol{z}$ has minimum Euclidean norm. Find a concise and computationally efficient expression for $\boldsymbol{z}$.

---

4. [3] Rewrite the following (simplified) MRI RF pulse design optimization problem in form that is solvable as a **linear programming** problem:

$$\underset{\boldsymbol{x} \in \mathbb{R}^N}{\arg\min} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_\infty \quad \text{such that} \quad \left\| \boldsymbol{x} \right\|_\infty \leq b,$$

   where $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ and $\boldsymbol{y} \in \mathbb{R}^M$ and $b > 0$. Alternatively you can think of this as a filter design problem where $\boldsymbol{y}$ denotes the desired frequency response, $\boldsymbol{x}$ denotes the filter coefficients, $\boldsymbol{A}$ is some form of Fourier transform, and b denotes a constraint on the maximum coefficient amplitude.

---

5. [9] Let $\mathcal{C}$ denote a **convex set** in $\mathbb{F}^N$. Prove, or disprove by counter example, the following statements about **convexity**.
   (a) [3] The **indicator function** $f(\boldsymbol{x}) = \mathbb{I}_{\{\boldsymbol{x} \in \mathcal{C}\}}$ is a convex function on $\mathbb{F}^N$.
   (b) [3] The function $g(\boldsymbol{x}) = 1 - \mathbb{I}_{\{\boldsymbol{x} \in \mathcal{C}\}}$ is a convex function on $\mathbb{F}^N$.
   (c) [3] The **characteristic function** $h(\boldsymbol{x}) = \chi_{\mathcal{C}}(\boldsymbol{x})$ is a convex function on $\mathbb{F}^N$.

   Hint. Consider the algebraic properties of the **extended real numbers**.

---

6. [9]

(a) [3] Let $\boldsymbol{T}$ denote a $K \times N$ matrix, and define the **regularizer**

$$R(\boldsymbol{x}) \triangleq \frac{1}{2} \min_{\boldsymbol{z} \in \mathbb{F}^K} \|\boldsymbol{T}\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \alpha \|\boldsymbol{z}\|_0 .$$

This expression, with its inner minimization over $\boldsymbol{z}$, can be convenient for **alternating minimization** methods, but it is also useful to have an equivalent expression without any such inner minimization.
In fact, we can express this regularizer directly as

$$R(\boldsymbol{x}) = \sum_{k=1}^{K} \psi([\boldsymbol{T}\boldsymbol{x}]_k),$$

for some **potential function** $\psi : \mathbb{C} \mapsto \mathbb{R}_+$. Determine $\psi(\cdot)$.
Hint. The function is sometimes called a "broken parabola" but that term is not easy to search online.

(b) [3] Now generalize the previous part to the more general regularizer

$$R(\boldsymbol{x}) = \min_{\boldsymbol{z} \in \mathbb{F}^K} \Phi(\boldsymbol{T}\boldsymbol{x} - \boldsymbol{z}) + \alpha \|\boldsymbol{z}\|_0 ,$$

for some function $\Phi : \mathbb{C}^K \mapsto \mathbb{R}_+$ that has the separable form $\Phi(\boldsymbol{z}) = \sum_{k=1}^{K} \phi(z_k)$, where convex function $\phi : \mathbb{C} \mapsto \mathbb{R}_+$ satisfies $\phi(z) = \phi(|z|)$ for all $z \in \mathbb{C}$ and $\phi(0) = 0$.

(c) [3] A related regularizer defined on $\mathbb{F}^N \times \mathbb{F}^K$ is

$$R(\boldsymbol{x}, \boldsymbol{z}) \triangleq \frac{1}{2} \|\boldsymbol{T}\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \alpha \|\boldsymbol{z}\|_1 .$$

Verify for yourself (do not submit) that $R(\boldsymbol{x}, \boldsymbol{z})$ is **convex** in $\boldsymbol{x}$ and in $\boldsymbol{z}$ individually (holding the other variable fixed). Now prove that $R(\boldsymbol{x}, \boldsymbol{z})$ is **jointly convex** in $\boldsymbol{x}$ and $\boldsymbol{z}$ on $\mathbb{F}^{N+K}$, or provide a counter-example for some $\boldsymbol{T}$.

---

7. [6]

Let $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{bmatrix}$. The spectral norm of $\boldsymbol{C}$ is relevant to finding the **Lipschitz constant** of some regularized cost functions. For practical use, we want easily computed bounds that scale to large problem sizes. Here are some practical upper bounds for the spectral norm of $\boldsymbol{C}'\boldsymbol{C}$:

- $\|\boldsymbol{C}\|_2^2 = \|\boldsymbol{C}'\boldsymbol{C}\|_2 \leq \|\boldsymbol{C}\|_1 \|\boldsymbol{C}\|_\infty \triangleq L_1$
- $\|\boldsymbol{C}\|_2^2 = \|\boldsymbol{C}'\boldsymbol{C}\|_2 = \|\boldsymbol{A}'\boldsymbol{A} + \boldsymbol{B}'\boldsymbol{B}\|_2 \leq \|\boldsymbol{A}'\boldsymbol{A}\|_2 + \|\boldsymbol{B}'\boldsymbol{B}\|_2 = \|\boldsymbol{A}\|_2^2 + \|\boldsymbol{B}\|_2^2 \leq \|\boldsymbol{A}\|_1 \|\boldsymbol{A}\|_\infty + \|\boldsymbol{B}\|_1 \|\boldsymbol{B}\|_\infty \triangleq L_2$
- $\|\boldsymbol{C}\|_2^2 \leq (\|\boldsymbol{A}\|_2 + \|\boldsymbol{B}\|_2)^2 = \left(\sqrt{\|\boldsymbol{A}\|_1 \|\boldsymbol{A}\|_\infty} + \sqrt{\|\boldsymbol{B}\|_1 \|\boldsymbol{B}\|_\infty}\right)^2 \triangleq L_3$

(a) [3] Show that $L_1 \leq (\|\boldsymbol{A}\|_1 + \|\boldsymbol{B}\|_1) \max(\|\boldsymbol{A}\|_\infty, \|\boldsymbol{B}\|_\infty) \triangleq L_4$
Smaller Lipschitz constants are preferable, so L_1 is preferable to L_4 when it is feasible to use it.

(b) [3] Show that $L_2 \leq L_3$, so L_2 is preferable to L_3

(c) [0] Challenge. Find an inequality relating L_1 and L_2, or show that none exists in general.

8. [41]
   (a) [3] Simplify the **elastic net** optimization problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_1 + \alpha \frac{1}{2} \|\boldsymbol{x}\|_2^2$$

   into the form of the **LASSO** optimization problem, so that if you had code for solving the LASSO problem you could apply it to the elastic net problem simply by passing in appropriate arguments to the LASSO solver.

   (b) [3] Determine an analytical solution to the above elastic net optimization problem when $\boldsymbol{A}$ is a **unitary matrix**.

   (c) [10] Write a JULIA function `enetu` that solves the elastic net optimization problem given $\boldsymbol{A}$, $\boldsymbol{y}$, α, β, when $\boldsymbol{A}$ is unitary. Your file should be named `enetu.jl` and should contain the following function:

```
"""
    xh = enetu(A, y, r1::Real, r2::Real)

Compute solution to elastic net regularization problem
`argmin_x 1/2 |A x - y|^2 + r1 |x|_1 + r2/2 |x|_2^2`
in the special case where `A` is a unitary matrix.
(The caller must ensure that `A` is unitary.)
Do not use any "for" loop in your solution!

In
* `A` `N x N` unitary matrix
* `y` vector of length N`
* `r1` l1 regularization parameter
* `r2` l2 regularization parameter

Out
* `xh`  solution to minimization problem (same size as y)
"""
function enetu(A, y, r1::Real, r2::Real)
```

   Submit your solution to mailto:eecs556@autograder.eecs.umich.edu.

   (d) [3] Apply your JULIA function to denoise a sinusoidal signal using the following code.

```
using LinearAlgebra: norm
using Statistics: mean
using Random: seed!
using Plots
include("enetu.jl") # include your function

N = 128
A = 1/sqrt(N) * [exp(-2im * pi * k * n / N) for k=0:N-1, n=0:N-1]
xfun = n -> 5 * cos(2*pi*7*n/N)
n = 1:N
xtrue = xfun.(n)
seed!(0)
y = xtrue + randn(N)
zh = enetu(A, y, 0.9, 0.05)
xh = real.(A * zh)
@show norm(y - xtrue)/norm(xtrue) # NRMSE of original data
@show norm(xh - xtrue)/norm(xtrue) # NRMSE after denoising
```

   Mathematically, this code is solving the following optimization problem for unitary DFT matrix $\boldsymbol{A}$:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{A}'\boldsymbol{x}\|_1 + \alpha \frac{1}{2} \|\boldsymbol{A}'\boldsymbol{x}\|_2^2$$

$$\hat{\boldsymbol{x}} = \boldsymbol{A}\hat{\boldsymbol{z}}, \quad \hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{z}\|_1 + \alpha \frac{1}{2} \|\boldsymbol{z}\|_2^2,$$

which is a reasonable approach when the DFT of $\boldsymbol{x}$ is sparse.
Report (on gradescope) the two **normalized root mean squared error** (**NRMSE**) values returned by this code.

(e) [3] Make a single figure that shows the original sinusoidal signal `xtrue`, the noisy signal `y` and the denoised signal `xh` (on the same axes) and submit a screen shot of it to gradescope.

(f) [3] Plot the magnitude DFT spectra of the noisy signal and of the denoised signal to see how they compare.
Hint: `abs.(A*y)` is the magnitude DFT spectrum of `y`.

(g) [3] Express the spectral norms of $\begin{bmatrix} \boldsymbol{A} \\ -\boldsymbol{A} \end{bmatrix}$ and $\begin{bmatrix} \boldsymbol{A} & -\boldsymbol{A} \end{bmatrix}$ in terms of the spectral norm of $\boldsymbol{A}$.
Hint. One of these will be useful in a later subproblem.

(h) [10] The **gradient projection** (GP) method, aka **projected gradient descent** method, for minimizing a function Ψ having smooth gradient with Lipschitz constant L, subject to the constraint $\boldsymbol{x} \in \mathcal{C}$ for a convex set $\mathcal{C}$, is given in general by

$$\boldsymbol{x}_{k+1} = \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{x}_k - \frac{1}{L}\nabla \Psi(\boldsymbol{x}_k)\right).$$

Following the course notes where $\boldsymbol{x} = \max(\boldsymbol{x}, \boldsymbol{0}) - \max(-\boldsymbol{x}, \boldsymbol{0})$, write a GP algorithm for solving the **LASSO problem** as a constrained least-squares problem. Write a JULIA function that performs this iterative algorithm given $\boldsymbol{A}$, $\boldsymbol{y}$, β and optionally an initial guess $\boldsymbol{x}_0$. Your function should have an optional argument where the caller can specify the step size. If the caller does not specify the step, then your function should use the default which is the reciprocal of the relevant Lipschitz constant. Think carefully about what L should be here!
Your function should take an optional named argument `fun` for evaluating `fun(x)` each iteration, and return a tuple of the final $\boldsymbol{x}$ and a vector of length `niter+1` with the values $\text{fun}(\boldsymbol{x}_0), \ldots, \text{fun}(x_{\text{niter}})$. If `fun` is not provided, then the second output argument is a vector of `undef` values. Initialize the `out` array to `undef` values by using:
`out = Array{Any}(undef, niter+1)`
All of the iterative algorithms in this course will need this optional user function argument so that you can compute quantities like $\Psi(\boldsymbol{x}_k)$ and $\|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\|_2$ while the iterative algorithm is running.
Your file should be named `lasso_cls.jl` and should contain the following function:

```
"""
  xh,out = lasso_cls(A, y, reg ; niter, x0, step, fun)

Iterative algorithm for the LASSO problem
`argmin_x 1/2 |A x - y|^2 + reg |x|_1`

Uses the constrained least-squares approach and gradient projection method
where we write `x = u - v = max(x,0) - max(-x,0)` and `|x|_1 = 1'u + 1'v`
Only suitable when `A` and `y` are real so that `x` is also real!

In
* `A` `M x N` real matrix
* `y::AbstractVector{<:Real}` vector of length `M`
* `reg::Real` regularization parameter

Option
* `x0::AbstractVector{<:Real}` initial guess (default A'y)
* `niter::Integer` number of iterations (default 10)
* `step::Real` step size (default 0 means use `1/Lipschitz`)
* `fun::Function` user-defined function evaluated at `x0` and at each iteration
default: `x -> undef`

Out
* `xh` "solution to" minimization problem after `niter` iterations
* `out` `[fun(x0), fun(x1), ..., fun(x_niter)]`
"""
function lasso_cls(A, y::AbstractVector{<:Real}, reg::Real ;
    niter::Integer=10,
    x0::AbstractVector{<:Real} = A'y,
```

```
    step::Real=0,
    fun::Function = x->undef)
```

Submit your solution to mailto:eecs556@autograder.eecs.umich.edu.

(i) [3] Apply your formula from (a) and use your `lasso_cls` function to solve (iteratively) the **elastic net** regularized LS problem for the following data and parameters.

```
N = 99
seed!(0)
A = svd(rand(N,N)).U
xtrue = randn(N)
y = xtrue + randn(N)
xh = enetu(A, y, 2.5, 0.2)
```

Plot $\log(\|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\| \,/\, \|\hat{\boldsymbol{x}}\|)$ vs k for $k = 0, \ldots, 20$, where $\hat{\boldsymbol{x}}$ is the (noniterative) solution from part (c). Initialize with $\boldsymbol{x}_0 = \boldsymbol{0}$. Of course it is not necessary to use an iterative method for this case where $\boldsymbol{A}$ is unitary, but using the unitary case where we know $\hat{\boldsymbol{x}}$ is a good way to test your LASSO / elastic net code.
Later you will use this same LASSO code for other cases where $\boldsymbol{A}$ is not unitary, and compare to other iterative methods for solving LASSO problems.