CSCI 6961/4961: Homework 3

Assigned Friday October 8 2021. Due by 11:59pm Thursday October 21 2021.

Create a Jupyter notebook for this assignment, and use Python 3. Write documented, readable and clear code (e.g. use reasonable variable names). Submit this notebook interspersing any textual answers in markdown cells (using LaTeX), clearly labeled, along with your code.

1. [10 pts] Show that the soft-shrinkage function on $\mathbb{R}$, defined as

$$\tau_\alpha(x_0) = \text{argmin}_{x\in\mathbb{R}} \frac{1}{2\alpha}(x - x_0)^2 + |x|$$

is well-defined (there is a unique minimizer), and use Fermat's optimality condition to conclude that

$$\tau_\alpha(x_0) = \left.\begin{cases} x_0 - \alpha & x_0 > \alpha \\ 0 & x_0 \in [-\alpha, \alpha] \\ x_0 + \alpha & x_0 < -\alpha \end{cases}\right\} = (|x_0| - \alpha)_+ \text{sign}(x_0).$$

2. [10 pts] Use the previous result to argue that the soft-shrinkage operator on vectors, defined by the solution to a convex optimization problem,

$$S_\alpha(\mathbf{x}_0) = \text{argmin}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}_0\|_2^2 + \|\mathbf{x}\|_1,$$

is well-defined (there is a unique minimizer), and that

$$S_\alpha(\mathbf{x}_0)_i = \tau_\alpha((\mathbf{x}_0)_i).$$

Write a function `softShrink(x0, alpha)` that implements the soft-shrinkage operator (efficiently, so no for loops).

3. [10 pts] Consider the LASSO problem

$$\mathbf{x}^\star = \text{argmin}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1. \tag{LASSO}$$

Write the expression for $\mathbf{x}_{t+1}$ in the composite gradient descent method[1] for solving (LASSO), using the soft-shrinkage operator. Assume that stepsizes α_t are given.

Write a function `[xT, objhist] = istaLasso(A, b, lambda_reg, x0, alpha, T)` that returns x_T, the Tth iterate of the composite gradient method for (LASSO) when the initial iterate is `x0` and the stepsize constant and given by `alpha`; it should also return a vector containing the objective values at each of the iterates $\mathbf{x}_0, \ldots, \mathbf{x}_T$. Use your function `softShrink`.

4. [10 pts] Compute the subdifferential of the objective of (LASSO) at a point $\mathbf{x}$.

Write a function `g = lassoSubgrad(A, b, lambda_reg, x)` that returns a subgradient for the LASSO objective at `x`.

Write a function `[xT, objhist] = subgradLasso(A, b, lambda_reg, x0, alpha, T)` that returns the last iterate of the subgradient descent method for solving (LASSO) when the initial iterate is `x0` and the stepsize is constant and given by `alpha`; it should also return a vector containing the value of the objective function at iterates $\mathbf{x}_0, \ldots, \mathbf{x}_T$. Use your function `lassoSubgrad`.

[1] See Participation 4, and take $f(\mathbf{x})$ to be the smooth part of the LASSO objective. The resulting algorithm is called an Iterative Shrinkage-Thresholding Algorithm, or ISTA.

5. [60 pts] Use the ISTA and subgradient solvers for the LASSO problem to solve the deblurring problem. The setup here is that we assume we observe a blurred and noisy version of an image, $\mathbf{b} = \mathbf{B}\mathbf{x} + \mathbf{e}$, and given knowledge of $\mathbf{B}$, our task is to recover an approximation of $\mathbf{x}$. Note that we are representing images as vectors by stacking the columns of the image together into a vector representation. Thus an image of size $n \times n$ is represented as a vector in $\mathbb{R}^{n^2}$.

   In general the blur matrix $\mathbf{B}$ is *not invertible*, but we could solve a least-squares problem to obtain an estimate of the image,

   $$\mathbf{x}_{\text{LS}} = \text{argmin}_{\mathbf{w}} \|\mathbf{B}\mathbf{w} - \mathbf{b}\|_2.$$

   This method doesn't work well if $\mathbf{B}$ is not close to invertible or in the presence of noise. In these cases, we want to take advantage of additional expectations on the image $\mathbf{x}$ to design a convex optimization problem whose solution is a better estimator of $\mathbf{x}$.

   In this exercise, we will use the popular modeling assumption that *natural images are approximately sparse in an appropriately chosen basis*. We will assume that there is a known orthonormal basis $\mathbf{H}$ in which $\mathbf{x}$ is approximately sparse: i.e., there is a vector $\mathbf{z}$ for which $\mathbf{x} = \mathbf{H}\mathbf{z}$, and $\|\mathbf{z}\|_1$ is small. This implies that $\mathbf{b} = \mathbf{B}\mathbf{H}\mathbf{z} + \mathbf{e} = \mathbf{A}\mathbf{z} + \mathbf{e}$, where $\mathbf{A} = \mathbf{B}\mathbf{H}$. Then we solve the LASSO problem

   $$\mathbf{z}^\star = \text{argmin}_{\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{z}\|_1$$

   for a judicious choice of λ and form the estimate $\mathbf{x}_{\text{LASSO}} = \mathbf{H}\mathbf{z}^\star$.

   A standard choice of $\mathbf{H}$ is given by the Haar wavelet basis, and for the data set and noise level you will use, I have found $\lambda = 2 \times 10^{-4}$ to give good results.

   - Download utility code for working with Haar matrices, blurring matrices, and converting between vectors and matrices from `https://www.cs.rpi.edu/~gittea/teaching/fall2021/files/deblurringUtils.py`.
   - Load the image we will be working with, resize it[2] so the problem is tractable, convert it to float format, and take a look at the image.

     ```
     from deblurringUtils import *
     from skimage import data

     n = 128
     camera = data.camera()
     smallcamera = resize(camera, [n, n])
     camdata = img_as_float(smallcamera)
     visualize(camdata)
     ```

   - Verify that the Haar basis does allow an almost sparse representation of the image

     ```
     H = vectorized2DHaarMatrix(n, n) # the Haar matrix, an orthonormal basis
     haarcoeffs = H.T @ vectorize(camdata) # convert to the Haar basis

     p = 90 # what percentage of the coefficients to set to zero
     thres = np.percentile(np.abs(haarcoeffs), p)
     haarcoeffs[abs(haarcoeffs) <= thres] = 0  # set most entries to zero

     # reconstruct the image from the sparse representation
     reconstim = unvectorize(H @ haarcoeffs, n, n)
     visualize(reconstim)
     ```

[2] You may use a smaller number during debugging, but n must be a power of two.

- Verify that a similarly sparse representation in the original pixel space is much worse.

```
import numpy.random as random

# mask the same percentage of pixels, randomly
mask = np.ones(n**2)
mask[:int(p/100*n**2)] = 0
mask = unvectorize(random.permutation(mask), n, n)
visualize(camdata * mask )
```

- Create the blurred, corrupted image. Corrupt using i.i.d. $\mathcal{N}(0, 1 \times 10^{-4})$ Gausian noise.

```
B = vectorized2DBlurMatrix(n, n, 5);
std = 1e-2
corruption = std * random.randn(n**2)
b = B @ vectorize(smallcamera) + corruption
blurredcam = unvectorize(b, n, n)
visualize(blurredcam)
```

- Compute and visualize the naive solution $\mathbf{x}_{\text{LS}}$.

```
from scipy.sparse.linalg import gmres
from scipy.sparse import csr_matrix

# solving with sparse matrices is faster
sB = csr_matrix(B)
linres, _ = gmres(sB, b, maxiter=50)
visualize(unvectorize(linres, n, n))
```

- Define the matrix $\mathbf{A}$ to be used in the LASSO problem, and compute the β-smoothness parameter of the smooth part of the LASSO objective, $\beta = \|\mathbf{A}^T\mathbf{A}\|_2$. Set the constant stepsize for both methods to $\alpha = \frac{1}{\beta}$.

```
from scipy.sparse.linalg import svds

A = B @ H
_, topsv, _ = svds(A, k=1)
alpha = 1/topsv**2
```

  What is the stepsize?
- Set $\lambda = 2 \times 10^{-4}$ as the regularization parameter and use 1000 steps of the subgradient solver to recover the image. Visualize the recovered image, and plot the objective values on a log-scale. What is the final objective value?
- Using the same value of λ, use 1000 steps of ISTA to recover the image. Visualize the recovered image, and plot the objective values on a log-scale. What is the final objective value?
- What conclusions do you draw about the relative merits of the two approaches to solving the LASSO problem? What are the advantages and disadvantages of solving the LASSO problem vs the least squares problem?
- We plotted the LASSO objective to verify visually that the methods seem to be converging. Comment on the meaningfulness of that value in measuring the *quality* of the recovered solution. Can you think of a more meaningful metric to report the quality of the iterates? (You may not use knowledge of the true image to compute that metric)

6. (CSCI 6961 students) [30 pts] Give a procedure for determining a useful value of λ in the LASSO formulation (you may not use knowledge of the true image in your procedure), and explain why you think this procedure will lead to a useful value of λ. Implement your procedure in the notebook, and use it to select a better value of λ than the one I selected by hand. Show the corresponding recovered image.