EECS 127/227AT Optimization Models in Engineering
Fall 2019 Homework 8

Release date: 10/24/19
Due date: 10/31/19, 23:00 (11 pm). Please LaTeX or handwrite your homework solution and submit an electronic version.

Submission Format
Your homework submission should consist of a single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

1. LASSO vs. Ridge

Say that we have the data set $\{(x^{(i)}, y^{(i)})\}_{i=1,\dots,n}$ of features $x^{(i)} \in \mathbb{R}^d$ and values $y^{(i)} \in \mathbb{R}$. Define $X = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}^\top$ and $y = \begin{bmatrix} y^{(1)} & \dots & y^{(n)} \end{bmatrix}^\top$. For the sake of simplicity, assume that the data has been centered and whitened so that each feature has mean 0 and variance 1 and the features are uncorrelated, i.e. $X^\top X = nI$.

Consider the linear least squares regression with regularization in the ℓ_1-norm, also known as LASSO:

$$w^* = \arg\min_{w \in \mathbb{R}^d} \ \|Xw - y\|_2^2 + \lambda \|w\|_1$$

This problem will compare ℓ_1-regularization with ℓ_2-regularization (ridge regression) to understand their similarities and differences. We will do this by looking at the elements of w^* in the solution to each problem.

(a) [Few lines of justification] First, decompose this optimization problem into d univariate optimization problems over each element of w. Let $X = \begin{bmatrix} x_1 & \dots & x_d \end{bmatrix}$.

(b) [Short justification, 1-2 lines] If $w_i^* > 0$, then what is the value of w_i^*? What is the condition on $y \top x_i$ for this to be possible?

(c) [Short justification, 1-2 lines] If $w_i^* < 0$, then what is the value of w_i^*? What is the condition on $y^\top x_i$ for this to be possible?

(d) [Short justification, 1-2 lines] What can we conclude about w_i^* if $|y^\top x_i| \leq \dfrac{\lambda}{2}$? How does the value of λ impact the individual entries w_i^*?

(e) [Few lines of justification] Now consider the case of ridge regression, which uses the the ℓ_2 regularization $\lambda \|w\|_2^2$.

$$w^* = \arg\min_{w \in \mathbb{R}^d} \ \|Xw - y\|_2^2 + \lambda \|w\|_2^2$$

What is the new condition for w_i^* to be 0? How does this differ from the condition obtained in part (d)? What does this suggest about LASSO?

2. Image Compression

Most of the work for this question will be done in `image_compression.ipynb`. The questions below are to help you write out the math that you will implement in code.

(a) [Single line expression] If $\mathbf{A} \in \mathbb{R}^{m \times n} = U\Sigma V^T$, what is a rank k approximation of $\mathbf{A}$ in terms of the columns and rows of U,Σ,V^T. Explicitly express the dimensions of your matrices.

(b) [Single line expression] We aim to learn a sparse representation $\hat{y} \in \mathbb{R}^{m \times n}$ of some data $y \in \mathbb{R}^{m \times n}$. Formulate this as a LASSO problem (using the Frobenius and L1 norms).

(c) Follow the instructions in `image_compression.ipynb`

3. Image Restoration

Finally! You did it!! After years of trying you finally got the perfect shot of the campanile. You were so excited you decided you should post it on Instagram. But first, you wanted to share it with your friends. You called them over and showed it to them. Your friends are from Stanford. They thought it would be cool to play a trick on you, so they edited your picture by adding the text " I wish I was at Stanford though! :(".

Figure 1: Your perfect shot corrupted by your Stanford friends :(

You see this in the morning, and realize you had no backup of your photo. While the prank was in good fun, you would like to get your original image back. So, you apply your Berkeley smarts in order to recreate the original image. You will be completing an exercise to try to restore the image your friends corrupted.

First, we will need to understand the representation an image (grayscale) stored in memory as a discrete sampling of a function over a continuous domain. One can think of an image as a function $f(x, y)$: $\Omega \subset \mathbb{R}^2 \to \mathbb{R}_+$. The range represents the color intensity at each point in the image.

To make the definition more concrete, we choose $\Omega = \{(x, y) : 1 \leq x \leq W+1, 1 \leq y \leq H+1\}$. Also in practice, the range of the function f may be restricted to the set $[0, 255]$, with 0 being black, 255 being white, and the numbers in between being various shades of gray going from dark to light.

When we store the image in memory, we sample this function at discrete points and store these samples (typically as a matrix). Given a fixed matrix with W rows and H columns. We may denote the sampled version of the function as: $F(i,j),\ \ i \in 1,2,\ldots,W,\ \ j \in 1,2,\ldots,H.$

(a) [Few lines of justification] To sample an image and ensure that we cover the domain, we may partition the domain into equal sized intervals and sample a point from each interval. One way to do this is to partition the interval of x values over the domain into intervals of width 1 and to partition the interval of y values of width 1. We may then sample points on the left boundary of each sub-interval. Given this assumption, explain why it makes sense that $F(i,j) = f\left(i,j\right),\ \ i \in 1,2,\ldots,W,\ \ j \in 1,2,\ldots,H.$

(b) [Single line expression] To solve our problem, it will be useful to approximate the gradient of the function f. Recall that the gradient $\nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right]$. For some small increment h, we may approximate it using:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{1}{h}(f(x+h,y) - f(x,y))$$
$$\frac{\partial f(x,y)}{\partial y} \approx \frac{1}{h}(f(x,y+h) - f(x,y))$$

Derive an expression for the gradient of F, $\nabla F(i,j) = G(i,j)$, based on F, for $i \in \{1,2,\ldots,W-1\},\ \ j \in \{1,2,\ldots,H-1\}$, using the approximation above. You may assume step-size $h = 1$.

(c) [Optimization Problem Expression] To solve our problem, we will also assume that we have an idea of which pixels i.e. (x,y) locations were corrupted by Oski's friends (indeed). We will denote this set $\mathcal{A}$, and it's discrete analog as A ((i,j) locations). One way to formulate the problem we are trying to solve is to try to create a new image f^* such that over all functions $\hat{f}\colon \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+$, f^* is the solution to:

$$\min_{\substack{\hat{f} \\ \hat{f}(x,y)=f(x,y) \\ \forall (x,y)\ \notin\ \mathcal{A}}} \int\int_{\Omega} ||\nabla \hat{f}(x,y)||_2 \ dx\, dy$$

Intuitively, the idea here is that we want to find the best approximation to the signal, by keeping the values where we know it is not corrupted, but allowing the values in corrupted areas to change, subject to those values not changing too much from their uncorrupted neighbors (hence the gradient term). What we should expect to happen is that the values at the corrupted pixels will be a smoothing of neighboring uncorrupted pixel. However, since we do not store the image as a continuous function, we will have to solve a discrete approximation to the problem above.
Denoting Ω as the region $a \le x \le b,\ \ c \le y \le d$ for scalars a,b,c,d, we may re-write the integral above as

$$\int_c^d \int_a^b ||\nabla \hat{f}(x,y)||_2 \ dx\, dy$$

If we partition the interval $[a,b]$ into m equal sub-partitions and $[c,d]$ into n equal sub-partitions, we may then approximate this integral by using a Riemann sum as

$$\sum_{i=1}^{m}\sum_{j=1}^{n} ||\nabla \hat{f}(x_i, y_j)||_2 \ \Delta x \, \Delta y$$

with $a = x_1, < x_2, \ldots, < x_{m+1} = b$, $c = y_1, < y_2, \ldots, < y_{n+1} = d$, $\Delta y = \frac{b-a}{m}$, $\Delta x = \frac{d-c}{n}$. Write explicitly a discrete approximation to the optimization problem above in terms of $\hat{F}, A, H$ and W. (You should ignore terms at the right-most boundaries of your estimate since, the gradient is not computed there. $\hat{G} = \nabla \hat{F} \in \mathbb{R}^{(H-1)\times(W-1)}$).

(d) [Optimization Problem Expression] Show that your formulation can be written as an SOCP (second order cone program).

(e) We will implement this in cvxpy. cvxpy has a function "cv.tv($\hat{F}$)", which when given a matrix $\hat{F}$, return the approximation of the objective function.

For the constraints we will take the following approach: Let the matrix A, be as follows:

$$A(i,j) = \begin{cases} 0 & \text{if } F(i,j) \text{ corrupt} \\ 1 & \text{otherwise} \end{cases}$$

Then we may write the constraint as $A \circ \hat{F} = A \circ F$, where $\circ$, known as the Hadamard product denotes element wise multiplication of the matrices.

This can be done in cvxpy as:

constraints = [cvxpy.multiply(A, $\hat{F}$) == cvxpy.multiply(A, F)].

Implement this in the Ipython notebook by completing the required portions of the code and test it on the sample images. The matrix A has already been supplied for you. Does is it work? Is there a scenario where this formulation would?

4. A slalom problem

A skier must slide from left to right by going through n parallel gates of known position (x_i, y_i) and width c_i, $i = 1, \ldots, n$. The initial position (x_0, y_0) is given, as well as the final one, (x_{n+1}, y_{n+1}). Before reaching the final position, the skier must go through gate i by passing between the points $(x_i, y_i - c_i/2)$ and $(x_i, y_i + c_i/2)$ for each $i \in \{1, \ldots, n\}$. See Figure 2.

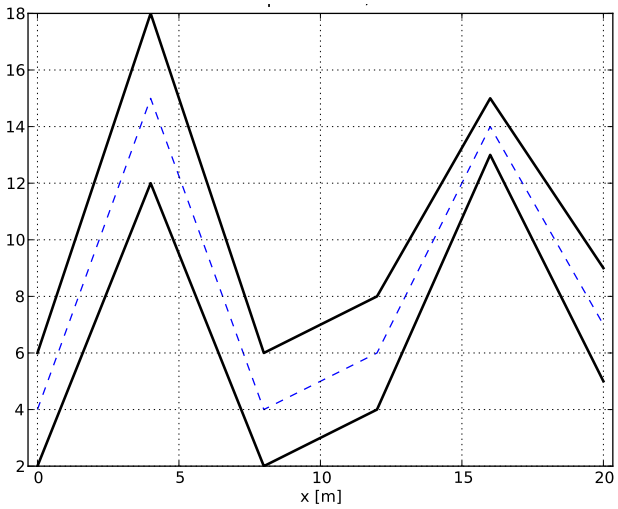

Figure 2: Slalom problem with $n = 6$ gates. The initial and final positions are fixed and not included in the figure. The skier slides from left to right. The middle path is dashed and connects the center points of gates.

Table 1: Problem data for Problem 2.

i	x_i	y_i	c_i
0	0	4	N/A
1	4	5	3
2	8	4	2
3	12	6	2
4	16	5	1
5	20	7	2
6	24	4	N/A

(a) [Optimization Problem Expression] Given the data $\{(x_i, y_i, c_i)\}_{i=0}^{n+1}$, write an optimization problem that minimizes the total length of the path. Your answer should come in the form of an SOCP.

(b) Solve the problem numerically with the data given in Table 1.

5. Sphere enclosure

[Few lines of justification] Let B_i, $i = 1, \ldots, m$, be m Euclidean balls in $\mathbb{R}^n$, with centers x_i, and radii $\rho_i \geq 0$. We wish to find a ball B of minimum radius that contains all the B_i, $i = 1, \ldots, m$. Cast this problem as an SOCP.