EE 516: Mathematical Foundations of Machine Learning **Winter 2023**

Homework 4
Due: February 12, 2023, 11:59PM PT

Student Name: *Instructor Name: John Lipor*

Since this is an exam week, I'm giving you two extra days to complete the homework. However, Homework 5 will still be due on Friday, February 17, so it is best to complete this before the weekend if possible.

Problem 1 (4 pts each)

Let $f(t) = 0.5e^{0.8t}$, $t \in [0, 2]$.

(a) Suppose 16 exact measurements of $f(t)$ are available to you, taken at the times t in the array T generated using the command `T = linspace(0,2,16)`. Use MATLAB or PYTHON to generate 16 exact measurements $y_i = f(t_i)$, $i = 1, \ldots, 16$. Write a script to determine the coefficients of the least-squares polynomial approximation of the data for

- a polynomial of degree 15: $p_{15}(t)$;
- a polynomial of degree 2: $p_2(t)$.

Compare the resulting functions with the true function by plotting all three functions over a fine grid on the same figure. Turn in your plot and a sentence or two summarizing the results qualatatively.

(b) Now generate noisy measurements $y_i = f(t_i) + n_i$, where n_i can be generated as `n = randn(length(T))`. Write another script to determine the coefficients of the least-squares polynomial approximation of the data for

- a polynomial of degree 15: $p_{15}(t)$;
- a polynomial of degree 2: $p_2(t)$.

Compare the resulting functions with the true function by plotting all three functions over a fine grid on the same figure. Turn in your plot and a sentence or two summarizing the results qualatatively.

Problem 2 (5 pts each)

Let

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The resulting least-squares solution is

$$\hat{x} = A^+ b = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

(a) Consider a perturbation $E_1 = \begin{bmatrix} 0 & \delta \\ 0 & 0 \end{bmatrix}$ of A, where δ is a small positive number. Solve the perturbed version of the problem: $\hat{z} = \arg\min_z \|A_1 z - b\|_2^2$, where $A_1 = A + E_1$. What happens to $\|\hat{x} - \hat{z}\|_2$ as δ approaches 0?

(b) Now consider a perturbation $E_2 = \begin{bmatrix} 0 & 0 \\ 0 & \delta \end{bmatrix}$, where again δ is a small positive number. Solve the perturbed version of the problem: $\hat{z} = \arg\min_z \|A_2 z - b\|_2^2$, where $A_2 = A + E_2$. What happens to $\|\hat{x} - \hat{z}\|_2$ as δ approaches 0?

Problem 3 (5 pts each)

Recall the least-squares problem

$$\hat{x} = \arg\min_x \|Ax - b\|_2^2.$$

When A is large, computing the pseudoinverse can be computationally prohibitive due to the cost of the SVD operation. In such settings, it is common to apply an optimization tool called *gradient descent*, which follows the iteration

$$x_{k+1} = x_k - \mu \nabla f(x_k),$$

where μ is the step size and $\nabla f(x_k)$ is the direction of descent. In the case where $f(x) = \frac{1}{2}\|Ax - b\|_2^2$, we have

$$\nabla f(x) = A^T A x - A^T b.$$

Hence, the iteration given by

$$x_{k+1} = x_k - \mu A^T(Ax_k - b) \tag{1}$$

will minimize $\|Ax - y\|_2^2$ as the iteration $k \to \infty$ when $0 < \mu < 2/\sigma_1^2(A)$, where $\sigma_1(A)$ is the largest singular value of A. Note that the iteration has a *fixed point*, i.e., $x_{k+1} = x_k$ when

$$A^T(Ax_k - b) = 0,$$

which are exactly the normal equations. So any fixed point minimizes the least-squares cost function!

(a) Complete the function called `lsgd` (in files for download) that implements the above least-squares gradient descent algorithm described by (1). Test your code using the included `prob5a` file and report the output. What do these values measure?

(b) After your implementation is complete, use it to generate a plot of $\|x_k - \hat{x}\|_2$ as a function of k for A and b as generated in `prob5b`. Test your algorithm for $\mu = c/\sigma_1^2(A)$ for $c \in \{0.1, 1, 1.9, 2\}$ and turn in one plot with all four curves on it. What do you observe about the convergence as a function of the step size?

Problem 4 (5 pts)

In this final installment of the photometric stereo problem, you will use your code from the previous two homework assignments to reconstruct a 3D surface from 2D images. In Homework 3, you computed the normal vectors to a particular surface from images by formulating and solving a least-squares problem. From vector calculus, we can express these surface normal vectors as

$$n(x,y) = \frac{1}{\sqrt{1 + \left(\frac{\partial}{\partial x} f(x,y)\right)^2 + \left(\frac{\partial}{\partial y} f(x,y)\right)^2}} \begin{bmatrix} -\frac{\partial}{\partial x} f(x,y) \\ -\frac{\partial}{\partial y} f(x,y) \\ 1 \end{bmatrix}, \tag{2}$$

where $\frac{\partial}{\partial x} f$ and $\frac{\partial}{\partial y} f$ denote the partial derivatives of $f(x,y)$ with respect to x and y, respectively. From (2), we can compute the partial derivatives as

$$\frac{\partial}{\partial x} f(x,y) = -\frac{n_1(x,y)}{n_3(x,y)} \qquad \frac{\partial}{\partial y} f(x,y) = -\frac{n_2(x,y)}{n_3(x,y)}, \tag{3}$$

where

$$n(x,y) = \begin{bmatrix} n_1(x,y) & n_2(x,y) & n_3(x,y) \end{bmatrix}^T. \tag{4}$$

In Homework 2, you constructed the A matrix satisfying

$$\begin{bmatrix} \texttt{dfdx} \\ \texttt{dfdy} \end{bmatrix} = A\ \texttt{fxy},$$

where `dfdx` and `dfdy` denote the vectorized approximations of the partial derivatives and `fxy` was the vectorized approximation of $f(x, y)$. Using (3), we can compute `dfdx` and `dfdy` from our normal vectors, and using our A matrix, we can solve the following least-squares problem

$$\texttt{fxy} = \arg\min_{f \in \mathbb{R}^{mn}} \left\| \begin{bmatrix} \texttt{dfdx} \\ \texttt{dfdy} \end{bmatrix} - Af \right\|_2^2 \tag{5}$$

to obtain the surface corresponding to our normal vectors.

Your task is to incorporate the functions `compute_normals` and `first_diffs_2d_matrix` into the `catdemo` script and formulate (5) using the outputs. Once `A` and `b` are appropriately defined in the script, this will generate a surface view of the object, completing the project. Turn in this plot, which should match the below if using MATLAB (the surface in PYTHON is less sharp).

After $f(x, y)$ is estimated, one can separately generate a stereolithography file that can be rendered on a 3D display or printed by a 3D printer. The MATLAB code to generate such a file is also included.

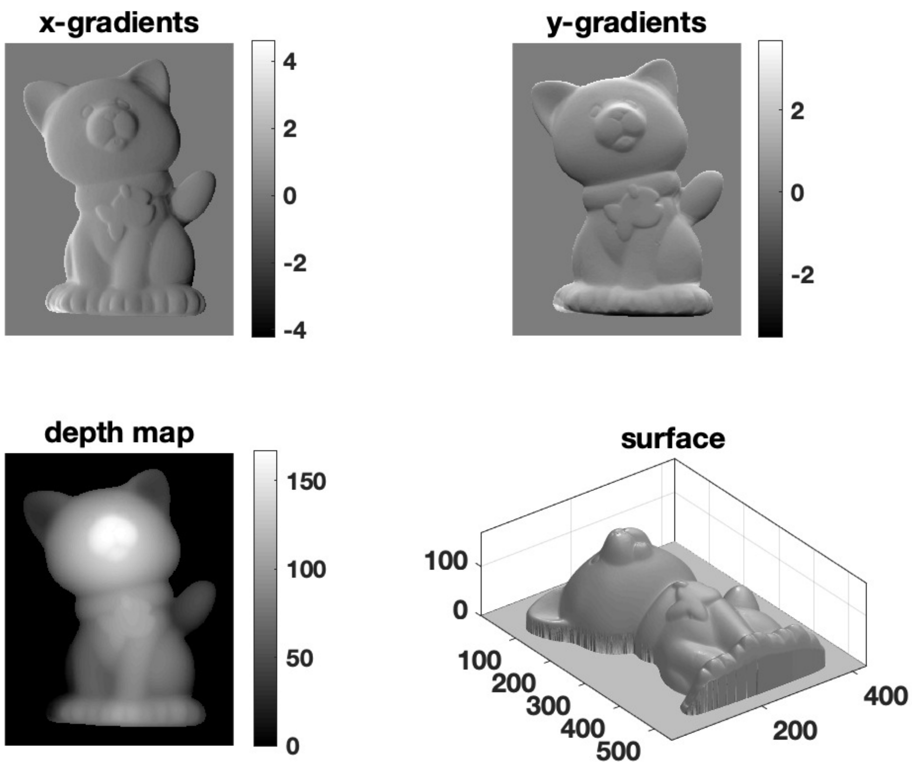

Figure 1: MATLAB figure generated from Problem 8.