Math 4650/MSSC 5650 - Homework 5

Instructor: Greg Ongie

Spring 2023

Instructions: To get started on Problems 1-4, unzip the folder `hw5` from the zip file `hw5.zip`. Inside the `hw5` folder you will find the scripts `problem1.m`, `problem2.m`, etc., which you should use to get started on each of the problems below. For these scripts to run you will need to have the *image processing toolbox* installed in MATLAB (to check if it is installed, type `ver` into the command line in MATLAB and see if "Image Processing Toolbox" shows up in the list of installed toolboxes; if not, install it by going to the "Apps" tab, and choose "Get More Apps"). Finally, be sure that the "current folder" in MATLAB as shown on the left-hand pane of the interface is set to the `hw5` folder, otherwise some scripts may not run correctly.

Problem 1 (10 pts MATH 4650 / 5 pts MSSC 5650). *Linear Regression.*
For this problem you will use linear regression to predict the date at which different bird species begin migration. Assume $a_1, ..., a_n$ are a collection of predictors that may be related to migration patterns for a given species, e.g., breeding latitude, wing-span, etc. Let b represent the median date of migration for that species, where $b = 1$ means April 1st, $b = 2$ means April 2nd and so on. We will fit a linear model of the form

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \approx b.$$

Here the x_i coefficients are unknown and can be learned from a collection of "training examples" of bird species where we know $a_1, ..., a_n$ and b. We will use 7 predictors $a_1, ..., a_7$ plus a constant predictor $a_8 = 1$; see `bird_migration.txt` for a description of these predictors.

Collecting all the predictor values for the different species into a matrix A and the responses into a vector b, the best fit coefficients $x = (x_1, x_2, ..., x_8) \in \mathbb{R}^8$ are the solution to the least squares problem:

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2.$$

Tasks:

(a) In the script `problem1.m`, find the regression coefficients $x \in \mathbb{R}^8$ that solve the above least squares problem (Note: A and b are already defined for you in the script). Include the code snippet you wrote (should just be one or two lines) in your write-up.

(b) Examine the regression coefficients you found in part (a). Based on the values of the coefficients, what predictors are positively correlated with a later migration date (i.e., which coefficients have a positive sign)? What predictors are negatively correlated with a later migration date (i.e., which coefficients have a negative sign)? What predictors have negligible impact on migration date (i.e., which coefficients are close to zero)? You may ignore the constant predictor a_8 in this analysis.

(c) Use the regression coefficients you found in part (a) to predict the migration date for a hold-out set of five species. The predictor matrix for these species is provided in the variable `A1`. How do your predictions compare to the known values given in the vector `b1`? Among the five species, what is the largest absolute difference between the predicted and actual migration date (in days)? What is the smallest absolute difference in migration date? And what is the mean absolute difference in migration date?

Problem 2 (7.5 pts MATH 4650 / 5 pts MSSC 5650)**.** *Image Denoising.*
A simple approach to denoising images is to use the following regularized least squares (RLS) formulation:

$$\min_{x\in\mathbb{R}^n} \|x-b\|^2 + \lambda\|Lx\|^2$$

where $b \in \mathbb{R}^n$ is the noisy image, $\lambda > 0$ is a parameter, and L is a matrix computes all finite differences in the horiztonal and vertical directions. This RLS problem has the solution

$$x_{RLS} = (I + \lambda L^\top L)^{-1} b.$$

Tasks:

(a) Implement the above formula for x_{RLS} in MATLAB in the script `problem2.m` (Note: I, L, and b are already defined for you in the script.) Include the code you wrote your report (should just be a few lines).

(b) Denoise the provided image for three choices of the regularizaiton parameter $\lambda > 0$:

- Choose λ such that little noise is removed.
- Choose λ such that too much noise is removed (image is over-smoothed).
- Choose λ that gives the best visual result, according to your eye.

Include the particular choices of λ and the resulting denoised images in your write-up.

Problem 3 (7.5 pts MATH 4650 / 5 pts MSSC 5650). *Image Deblurring.*
Image deblurring can be posed as a least squares problem

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|^2 + \lambda\|x\|^2$$

where $A \in \mathbb{R}^{n\times n}$ is the blur matrix, $b \in \mathbb{R}^n$ is the blurred image, $\lambda > 0$ is a parameter. This RLS problem has the solution

$$x_{RLS} = (A^\top A + \lambda I)^{-1} A^\top b.$$

Tasks:

(a) Implement the above formula for x_{RLS} in MATLAB in the script `problem3.m` (Note: A, I, and b are already defined for you in the script). Include the code snippet you wrote in your report (should just be a few lines).

(b) Deblur the provided image using three choices of the regularization parameter $\lambda \geq 0$:

- Choose $\lambda = 0$ (no regularization). What do you observe?
- Choose $\lambda > 0$ such that the image is sharper but there is too much noise.
- Choose λ that gives the best visual result, according to your eye.

Include the particular choices of λ and the resulting deblurred images in your report.

Problem 4 (10 pts MSSC 5650). *Inpainting Face Images with Eigenfaces.*
Let $\hat{U} \in \mathbb{R}^{n\times k}$ be the matrix whose columns represent the top-k eigenfaces, and let $y \in \mathbb{R}^n$ denote the mean face image. Recall that if $x \in \mathbb{R}^n$ is a face image the squared distance to face space is given by

$$d_{FS}^2(x) = \|(x-y) - \hat{U}\hat{U}^\top(x-y)\|^2 = \|(I - \hat{U}\hat{U}^\top)(x-y)\|^2.$$

This can be used as a regularization function for least squares problems involving face images, i.e., if $x \in \mathbb{R}^n$ is a face image and $b \in \mathbb{R}^m$ are measurements, and $A \in \mathbb{R}^{m\times n}$ is a given measurement matrix, we can consider the RLS problem

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|^2 + \lambda\|L(x-y)\|^2$$

where we have set $L = I - \hat{U}\hat{U}^\top$.

In this problem we will consider the task of "inpainting" face images that have missing pixels. In this case A is a diagonal matrix with 1's and 0's along the diagonal, where a 1 indicates the pixel is present, and 0 indicates the pixel is absent. See the script `problem4.m` for an example, where a rectangle of pixels over the eyes is missing. The goal is to fill in these missing pixels to make a plausible looking face.

Tasks:

(a) Derive a formula for the solution x_{RLS} to the RLS problem above for a general A matrix. Include this in your report.

(b) Implement your formula for x_{RLS} in MATLAB in the script `problem4.m` to solve the inpainting problem. (Note: A, I, $\hat{U}$, b, and y are defined for you in the script). Include the code snippet you wrote in your report (should just be a few lines).

(c) Use your implementation to inpaint the missing pixels of the provided face image. Choose the value of $\lambda > 0$ that gives the most plausible looking face image while not deviating too much from the observed pixels. Include the resulting inpainted image in your report.

Only for students that have taken MSSC 6040 Applied Linear Algebra with me. You do not need to redo Problems 1-4 if you have solved these problems previously in MSSC 6040. Solve only the following problem:

Problem 5 (25 pts)**.** Let A be an $m \times n$ real matrix with $m < n$. Assume A is full rank, i.e., $\text{rank}(A) = m$, and let b be any vector in $\mathbb{R}^m$. In this case, the system $Ax = b$ is underdetermined and has infinitely many solutions. The *least-norm solution* is defined as the solution of the system having minimum Euclidean norm, i.e., the minimizer of the optimization problem:

$$\min_{x \in \mathbb{R}^n} \|x\| \quad s.t. \quad Ax = b.$$

The following steps will have you investigate properties of the least-norm solution.

(a) Prove that $x_0 = A^\top (AA^\top)^{-1} b$ is a solution of the system $Ax = b$.

(b) Prove that if x is any vector such that $Ax = b$ and $x \in \text{range}(A^\top)$, then $x = x_0$.

(c) Prove that x_0 is the unique *least-norm solution*, i.e., show that if $x \in \mathbb{R}^n$ satisfies $Ax = b$, then $\|x\|_2 \geq \|x_0\|_2$, and this holds with strict inequality if $x \neq x_0$. (Hint: Note that $A(x - x_0) = 0$. Use this to show x_0 and $x - x_0$ are orthogonal. Now apply the Pythagorean theorem.)

(d) Consider a reduced SVD of A given as $A = U\hat{\Sigma}\hat{V}^\top$ where $U \in \mathbb{R}^{m\times m}$ is orthogonal, $\hat{\Sigma} \in \mathbb{R}^{m\times m}$ is the diagonal matrix containing the non-zero singular values of A, and $\hat{V} \in \mathbb{R}^{n\times m}$ has orthonormal columns. Prove that $x_0 = \hat{V}\hat{\Sigma}^{-1}U^\top b$.

(e) For any $\lambda > 0$, let x_λ be the solution to the regularized least squares problem:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2 + \lambda \|x\|^2.$$

Prove that $x_0 = \lim_{\lambda \to 0^+} x_\lambda$. (Hint: express both x_λ and x_0 in terms of a full SVD of A, and show $\|x_\lambda - x_0\| \to 0$ as $\lambda \to 0^+$.)

(f) Consider a reduced QR factorization of $A^\top$ given as $A^\top = \hat{Q}R$, where $\hat{Q} \in \mathbb{R}^{n\times m}$ has orthonormal columns, and $R \in \mathbb{R}^{m\times m}$ is upper triangular and nonsingular. Derive a simplified formula for the least-norm solution $x_0 = A^\top (AA^\top)^{-1} b$ in terms of the QR factors. Use this formula to describe an algorithm to compute the least-norm solution numerically (you do not need to describe how to compute a QR factorization).

(g) Modify the provided `datafit_demo.m` script in the following way: set the number of data points to $m = 3$ and the polynomial degree to $d = 5$, which results in an underdetermined system $Xc = y$ (i.e., there are many degree 5 polynomials fit these three data points). By default, the "backslash" command already in the script `c = X\y;` will find the least-norm solution and plot the resulting polynomial fit. Your task is to find the least-norm solution of $Xc = y$ using the computational approach suggested by part (f), and verify it matches the result of backslash. Include a plot of the polynomial fit in your write-up. Finally, investigate what happens to the polynomial fit for degrees $d = 6, 7, 8$. What do you observe?

Solution. (a) If $x_0 = A^\top(AA^\top)^{-1}b$ then $Ax_0 = AA^\top(AA^\top)^{-1}b$, since A is an $m \times n$ matrix with $rank(A) = m$, then $AA^\top$ is invertible. Therefore $AA^\top(AA^\top)^{-1} = I$ and $Ax_0 = AA^\top(AA^\top)^{-1}b = Ib = b$. Which mean that x_0 is a solution to the system $Ax = b$.

(b) Since $x \in range(A^\top)$, then there exists a vector y such that $A^\top y = x$.

Then the linear system is $Ax = b$ can be written as $AA^\top y = b$, and we know $AA^\top$ is invertible, then $y = (AA^\top)^{-1}b$, multiplying by $A^\top$ on the left, then we have

$$A^\top y = A^\top(AA^\top)^{-1}b$$

Where $A^\top y = x$ and $A^T(AA^\top)^{-1}b = x_0$, then $x = x_0$

(c) Since $Ax = b$ and $Ax_0 = b$, then

$$\begin{aligned} Ax &= Ax_0 \\ Ax - Ax_0 &= 0 \\ A(x - x_0) &= 0 \end{aligned}$$

Which means that $x - x_0$ is in the nullspace of A. For x_0 and $x - x_0$ to be orthogonal, we need to look at $x_0^\top(x - x_0)$.

$$\begin{aligned} x_0^\top(x - x_0) &= (A^\top(AA^\top)^{-1}b)^\top(x - x_0) \\ &= b^\top((AA^\top)^{-1})^\top A(x - x_0) \\ &= b^\top((AA^\top)^{-1})^\top(0) \\ &= 0 \end{aligned}$$

Since they are orthogonal, then we can use the Pythagorean theorem to compare the norm of x and x_0:

$$\|x\|_2^2 = \|x - x_0 + x_0\|_2^2 = \|x - x_0\|_2^2 + \|x_0\|_2^2 \geq \|x_0\|_2^2$$

This follows since $\|x - x_0\|_2^2 \geq 0$ and $\|x - x_0\|_2^2 = 0$ if and only if $x = x_0$

Therefore $\|x\|_2 \geq \|x_0\|_2$, and this holds with strict inequality if $x \neq x_0$.

(d) Since $A = U\hat{\Sigma}\hat{V}^\top$, then $A^\top = \hat{V}\hat{\Sigma}^\top U^\top$. Using the properties of U as an orthogonal matrix, and $\hat{\Sigma}$ is a diagonal matrix, and $\hat{V}$ having orthonormal columns, let's rewrite x_0

$$\begin{aligned}
x_0 &= \hat{V}\hat{\Sigma}^\top U^\top (U\hat{\Sigma}\hat{V}^\top \hat{V}\hat{\Sigma}^\top U^\top)^{-1} b \\
&= \hat{V}\hat{\Sigma}^\top U^\top (U\hat{\Sigma}\hat{\Sigma}^\top U^\top)^{-1} b \quad \text{using } \hat{V}^\top \hat{V} = I \\
&= \hat{V}\hat{\Sigma}^\top U^\top (U\hat{\Sigma}^2 U^\top)^{-1} b \quad \text{using } \hat{\Sigma} = \hat{\Sigma}^\top \\
&= \hat{V}\hat{\Sigma}^\top U^\top ((U^\top)^{-1}\hat{\Sigma}^{-2} U^{-1}) b \\
&= \hat{V}\hat{\Sigma}^\top U^\top ((U^\top)^\top \hat{\Sigma}^{-2} U^\top) b \quad \text{using } U^\top = U^{-1} \\
&= \hat{V}\hat{\Sigma} U^\top U \hat{\Sigma}^{-2} U^\top b \\
&= \hat{V}\hat{\Sigma}\hat{\Sigma}^{-2} U^\top b \\
&= \hat{V}\hat{\Sigma}^{-1} U^\top b
\end{aligned}$$

(e) The full SVD of $A = U\Sigma V^\top$, then let's look at x_λ

$$\begin{aligned}
x_\lambda &= [(U\Sigma V^\top)^\top (U\Sigma V^\top) + \lambda L^\top L]^{-1} (U\Sigma V^\top) b \\
&= [V\Sigma^\top U^\top U\Sigma V^\top + \lambda L^\top L]^{-1} (V\Sigma^\top U^\top) b \\
&= [V\Sigma^2 V^\top + \lambda L^\top L]^{-1} (V\Sigma U^\top) b \\
&= [(V\Sigma^2 V^\top)^{-1} + (\lambda L^\top L)^{-1}] (V\Sigma U^\top) b \\
&= [((V^\top)^{-1}\Sigma^{-2} V^{-1}) + (\lambda L^\top L)^{-1}] (V\Sigma U^\top) b \\
&= [(V\Sigma^{-2} V^\top) + (\lambda L^\top L)^{-1}] (V\Sigma U^\top) b \\
&= V\Sigma^{-2} V^\top V\Sigma U^T b + \lambda (L^\top L)^{-1} V\Sigma U^\top b \\
&= V\Sigma^{-1} U^\top b + \lambda (L^\top L)^{-1} V\Sigma U^\top b
\end{aligned}$$

And x_0

$$\begin{aligned}
x_0 &= (U\Sigma V^\top)^\top [(U\Sigma V^\top)(U\Sigma V^\top)^\top]^{-1} b \\
&= (V\Sigma U^\top)[(U\Sigma V^\top)(V\Sigma U^\top)]^{-1} b \\
&= V\Sigma U^\top (U\Sigma^2 U^\top)^{-1} b \\
&= V\Sigma U^\top U\Sigma^{-2} U^\top b \\
&= V\Sigma^{-1} U^\top b
\end{aligned}$$

Then let's look at the norm difference between x_0 and x_λ:

$$\begin{aligned}\|x_\lambda - x_0\|^2 &= \|V\Sigma^{-1}U^\top b + \lambda(L^\top L)^{-1}V\Sigma U^\top b - V\Sigma^{-1}U^\top b\|^2\\ &= \|\lambda(L^\top L)^{-1}V\Sigma U^\top b\|^2\end{aligned}$$

$$\begin{aligned}\Rightarrow lim_{\lambda\to0^+}\|x_\lambda - x_0\|^2 &= lim_{\lambda\to0^+}\|\lambda(L^\top L)^{-1}V\Sigma U^\top b\|^2\\ &= \|0(L^\top L)^{-1}V\Sigma U^\top b\|^2\\ &= 0\end{aligned}$$

Therefore $lim_{\lambda\to0^+}x_\lambda = x_0$, since the difference in the norm approaches 0 as $\lambda \to 0^+$

(f) Let's rewrite x_0 using the properties of $\hat{Q}$ and R:

$$\begin{aligned}x_0 &= \hat{Q}R((\hat{Q}R)^\top\hat{Q}R)^{-1}b\\ &= \hat{Q}R(R^\top\hat{Q}^\top R)^{-1}b\\ &= \hat{Q}R(R^\top R)^{-1}b\\ &= \hat{Q}RR^{-1}R^{-\top}b\\ &= \hat{Q}(R^\top)^{-1}b\end{aligned}$$

Then we can use the following algorithm to compute the least-norm solution.

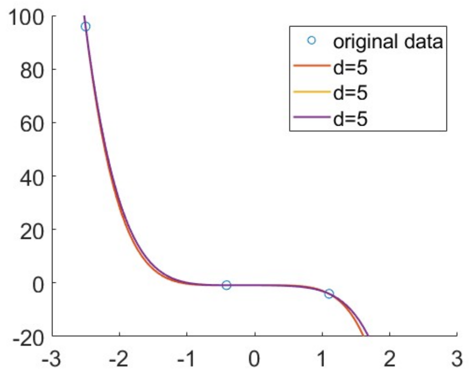
```
    1 - Compute the reduced QR
    2 - Compute the least norm solution by using the formula
        above and the QR factors
```

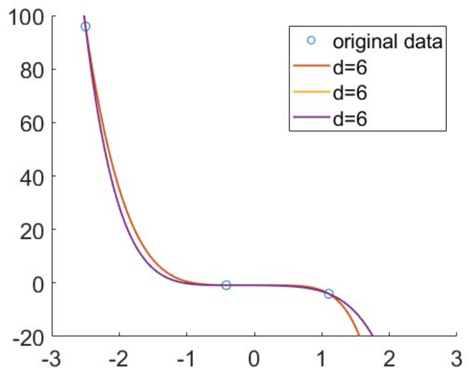
(g) .

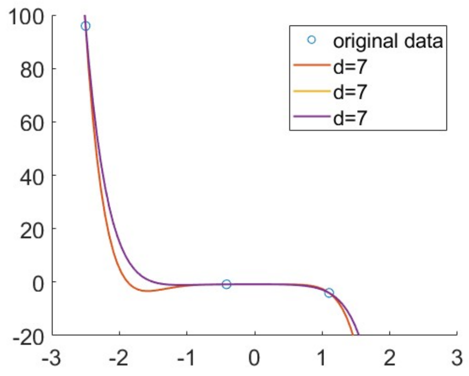
```
%% generate noisy samples from from 4th degree polynomial
rng(1); %fix random seed

m = 3; %number of data-points
f = @(x) 1 -2*x + x.^2 - 3*x.^3 + x.^4;
x = 2.5*(2*rand(m,1)-1); %sample from interval [-2.5,2.5]
y = f(x) + 2*randn(m,1); %noisy samples

figure(1);
scatter(x,y,50,'DisplayName','original data');
legend;
set(gca,'fontsize',18);
axis([-3 3 -20 100]);
```

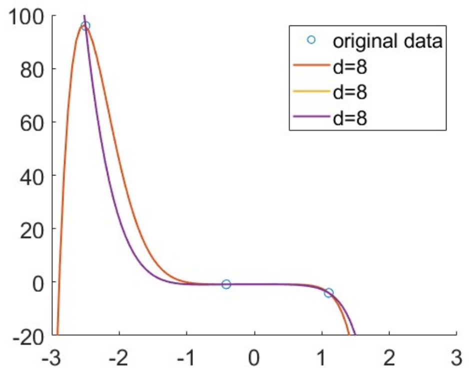
```
%% solve for coefficients c using backslash
d = 5;
X = vandermonde(x,d);
c = X\y;
c1 = leastNormSolution(X,y);
c2 = lsqminnorm(X,y);
% plot result on top of scattered data
xp = linspace(-3,3,100);
Xp = vandermonde(xp,d);
poly = Xp*c;
poly1 = Xp*c1;
poly2 = Xp*c2;

figure(1);
hold all;
plot(xp,poly, xp,poly1,xp,poly2,'DisplayName',sprintf('d=%d'
   ,d),'linewidth',1.5);
hold off;
legend;
set(gca,'fontsize',18);
axis([-3 3 -20 100]);

%% helper functions
function x0 = leastNormSolution(A, b)
    % Compute the reduced QR factorization of A'
    [Q, R] = qr(A',0);
    %Q = Q(:, 1:size(A, 1));
    %R = R(1:size(A, 1), :);

    % Compute the least-norm solution using the QR factors
    x0 = Q * (R'\b);
end

%create Vandermonde matrix X from data-points x_i
function X = vandermonde(x,d)
    %x = m x 1 vector of inputs x_i
    %d = polynomial degree
    X = [];
    for k=0:d
        X(:,end+1) = x.^k;
    end
end
```

And the plots, where yellow is my solution, purple is the solution given by the MATLAB function `lsqminnorm`, and the backslash solution is in orange:









Looking at the plots, it is possible to see that the solution using the least norm solution via QR seems more stable as the degrees increase, while the solution using the "backslash" operator gets more and more unstable as the degrees increase. It is possible to see that for degree 8 the curve would have a lot of errors if compared with the "true" function.

The reason why I used this other function to test against my own solution is because the "backslash" operator does something slightly different than the least norm solution. While this function computes the least norm solution. It is possible to see that both my solution and the MATLAB function give exactly the same result, while the "backslash" operator returns a different polynomial.

The following link explain the difference between the "backslash" and the lsqminnorm: https://www.mathworks.com/help/matlab/ref/lsqminnorm.html