Homework 5

Don’t forget to include a print copy of your Julia code where required.

1. Weighted least-squares. In least squares, the objective (to be minimized) is

\[ \|Ax - b\|^2 = \sum_{i=1}^{m} (\tilde{a}_i^T x - b_i)^2, \]

where \(\tilde{a}_i^T\) are the rows of \(A\), and the \(n\)-vector \(x\) is to chosen. In the weighted least-squares problem, we minimize the objective

\[ \sum_{i=1}^{m} w_i (\tilde{a}_i^T x - b_i)^2, \]

where \(w_i\) are given positive weights. The weights allow us to assign different weights to the different components of the residual vector.

(a) Show that the weighted least-squares objective can be expressed as \(\|D(Ax - b)\|^2\) for an appropriate diagonal matrix \(D\). This allow us to solve the weighted least-squares problem as a standard least-squares problem, by minimizing \(\|\tilde{A}x - \tilde{b}\|^2\), where \(\tilde{A} = DA\) and \(\tilde{b} = Db\).

(b) Show that when \(A\) has independent columns, so does the matrix \(\tilde{A}\).

(c) The least-squares approximate solution is given by \(\hat{x} = (A^T A)^{-1} A^T b\). Give a similar formula for the solution of the weighted least-squares problem. You might want to use the matrix \(W = \text{diag}(w)\) in your formula.

2. Solving least-squares problems in Julia. Generate a random \(20 \times 10\) matrix \(A\) and a random \(20\)-vector \(b\).

(a) Compute the solution \(\hat{x}\) of the associated least-squares problem using the methods listed below, and verify that the solutions found are the same, or more accurately, very close to each other; they will be very slightly different due to small roundoff errors in the computations.

- Using the Julia backslash operator.
- Using \(\hat{x} = (A^T A)^{-1} A^T b\).
- Using \(\hat{x} = A^\dagger b\).

Hints. In Julia, \(\text{inv()}\) computes the inverse matrix, \(\text{pinv()}\) computes the pseudo-inverse matrix, and \(A\backslash b\) directly solves the least-squares problem.
(b) Let $\hat{x}$ be one of the solutions found in part (a). Generate a random nonzero $n$-vector $\delta$ and verify that $\|A(\hat{x} + \delta) - b\|^2 > \|A\hat{x} - b\|^2$. Repeat several times with different values of $\delta$; you might try choosing a small $\delta$ (say, by scaling the original random vector).

Be sure to submit your code, including the code that checks if the solutions in part (a) are close to each other, and whether the expected inequality in part (b) holds.

3. **Julia timing test for least-squares.** Determine how long it takes for your computer to solve a least-squares problem with $m = 100000$ equations and $n = 100$ variables. (You can use the backslash operator.)

   *Remark.* Julia compiles just in time, so you should run the code a few times to get the correct time.

4. **Saving TA time using midterm score prediction.** The TAs very carefully graded all problems on all students' midterms. But suppose they had just graded the first half of the exam, i.e., problems 1–5, and used a regression model to predict the total score on each exam.

   The $95 \times 10$ matrix of midterm scores is available online, but for your convenience, we have created the data file `midterm_scores.jl` for you. This file can be loaded with `include("midterm_scores.jl")`:

   (a) Find the average and standard deviation of the total midterm scores.

   (b) Here is a very simple way to predict the total score based on the 5 scores for first half of the exam: Sum the 5 first half scores, and double the result. What RMS prediction error does this simple method achieve?

   (c) Find a regression model that predicts the total score based on the 5 scores for problems 1–5. Give the coefficients and briefly interpret them. What is the RMS prediction error of your model?

      (Just for fun, evaluate the predictor on your own exam score. Would you rather have your actual score or your predicted score?)

   *Remark.* Your dedicated EE103 teaching staff would *never* do anything like this. Really.

5. **Moore’s law.** The figure and table below show the number of transistors $N$ in 13 microprocessors, and the year of their introduction.
<table>
<thead>
<tr>
<th>Year</th>
<th>Number of Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1971</td>
<td>2,250</td>
</tr>
<tr>
<td>1972</td>
<td>2,500</td>
</tr>
<tr>
<td>1974</td>
<td>5,000</td>
</tr>
<tr>
<td>1978</td>
<td>29,000</td>
</tr>
<tr>
<td>1982</td>
<td>120,000</td>
</tr>
<tr>
<td>1985</td>
<td>275,000</td>
</tr>
<tr>
<td>1989</td>
<td>1,180,000</td>
</tr>
<tr>
<td>1993</td>
<td>3,100,000</td>
</tr>
<tr>
<td>1997</td>
<td>7,500,000</td>
</tr>
<tr>
<td>1999</td>
<td>24,000,000</td>
</tr>
<tr>
<td>2000</td>
<td>42,000,000</td>
</tr>
<tr>
<td>2002</td>
<td>220,000,000</td>
</tr>
<tr>
<td>2003</td>
<td>410,000,000</td>
</tr>
</tbody>
</table>

These numbers are available in the Julia file `moore_data.jl`. In this data file, $t$ is the first column of the table (introduction year) and $N$ is the second column (number of transistors).

The plot gives the number of transistors on a logarithmic scale. Find the least-squares straight-line fit of

$$(\log_{10} N, t - 1970),$$

where $t$ is the year and $N$ is the number of transistors, from the given data. This gives an approximation of the form

$$\log_{10} N \approx \theta_1 + \theta_2 (t - 1970).$$

(a) Find the coefficients $\theta_1$ and $\theta_2$ that minimize the RMS error on the data, and give the RMS error on the data. Plot the model you find along with the data points.

(b) Use your model to predict the number of transistors in a microprocessor introduced in 2015. Compare the prediction to the IBM Z13 microprocessor, released in 2015, which has around $4 \times 10^9$ transistors.

(c) Compare your result with Moore’s law, which states that the number of transistors per integrated circuit roughly doubles every one and a half to two years.

**Hints.** In Julia, the function $\log_{10}$ is `log10`.

6. **Orthogonality principle.**

(a) Suppose that $A$ has independent columns, and $\hat{x}$ is the least-squares approximate solution of $Ax = b$. Let $\hat{r} = A\hat{x} - b$ be the associated residual. Show that for any $x$, $(Ax) \perp \hat{r}$. This is sometimes called the orthogonality principle: every linear combination of the columns of $A$ is orthogonal to the least-squares residual.
(b) Now consider a data fitting problem, with first basis function \( \phi_1(x) = 1 \), and data set \((x_1, y_1), \ldots, (x_N, y_N)\). Assume the matrix \( A \) in the associated least-squares problem has independent columns, and let \( \hat{\theta} \) denote the parameter values that minimize the mean square prediction error over the data set. Let the \( N \)-vector \( \hat{r} \) denote the prediction errors using the optimal model parameter \( \hat{\theta} \). Show that \( \text{avg}(\hat{r}) = 0 \). In other words: With the least-squares fit, the mean of the prediction errors over the data set is zero.

Hint. Consider part (a), with \( x = e_1 \).

7. Some basic properties of convolution. Suppose that \( a \) is an \( n \)-vector, \( b \) is an \( m \)-vector, and \( c \) is a \( p \)-vector.

(a) Show that convolution is commutative, \( i.e., \ a \ast b = b \ast a \).

Hint. Use the formula \((a \ast b)_k = \sum_{i+j=k+1} a_i b_j\), where the sum means you should sum over all \( i, j \) that satisfy \( i + j = k + 1 \), and we interpret \( a_i \) and \( b_j \) as zero when the index is outside its normal range.

(b) Show that convolution is associative, \( i.e., \ (a \ast b) \ast c = a \ast (b \ast c) \). (This justifies writing either of these as \( a \ast b \ast c \).)

Hint. Show that both the left and right hand sides can be expressed as

\[
(a \ast b \ast c)_k = \sum_{i+j+\ell=k+2} a_i b_j c_\ell,
\]

with \( a_i, b_j, \) and \( c_\ell \) interpreted as 0 outside their ranges.

(c) Convolution with 1. What is \( 1 \ast a \)? (Here we interpret 1 as a 1-vector.)

(d) Convolution with a unit vector. Let \( e_{k,q} \) denote the \( k \)th unit vector of dimension \( q \). What is \( e_{k,q} \ast a \)? Describe this vector mathematically \( (i.e., \ give \ its \ entries) \), and via a brief English description. You might find vector slice notation useful.

8. Convolution in Julia. Use Julia’s \texttt{conv()} \ function to find the coefficients of the polynomial \((1 - x + 2x^2)^4\). \textit{Hint.} Convolution gives the coefficients of the product of two polynomials.