Homework 3

EE 263 Stanford University Summer 2018

Due: July 18, 2018

1. Least-squares residuals. Suppose A is skinny and full-rank. Let x_{ls} be the least-squares approximate solution of $Ax = y$, and let $y_{\text{ls}} = Ax_{\text{ls}}$. Show that the residual vector $r = y - y_{\text{ls}}$ satisfies

$$\|r\|^2 = \|y\|^2 - \|y_{\text{ls}}\|^2.$$

Also, give a brief geometric interpretation of this equality (just a couple of sentences, and maybe a conceptual drawing).

2. Least-squares model fitting. In this problem you will use least-squares to fit several different types of models to a given set of input/output data. The data consist of a scalar input sequence u, and a scalar output sequence y, for $t = 1, \ldots, N$. You will develop several different models that relate the signals u and y.

- *Memoryless models.* In a memoryless model, the output at time t, *i.e.*, $y(t)$, depends only the input at time t, *i.e.*, $u(t)$. Another common term for such a model is *static*.

$$\begin{aligned}
\text{constant model:} &\quad y(t) = c_0 \\
\text{static linear:} &\quad y(t) = c_1 u(t) \\
\text{static affine:} &\quad y(t) = c_0 + c_1 u(t) \\
\text{static quadratic:} &\quad y(t) = c_0 + c_1 u(t) + c_2 u(t)^2
\end{aligned}$$

- *Dynamic models.* In a dynamic model, $y(t)$ depends on $u(s)$ for some $s \neq t$. We consider some simple time-series models (see problem 2 in the reader), which are linear dynamic models.

$$\begin{aligned}
\text{moving average (MA):} &\quad y(t) = a_0 u(t) + a_1 u(t-1) + a_2 u(t-2) \\
\text{autoregressive (AR):} &\quad y(t) = a_0 u(t) + b_1 y(t-1) + b_2 y(t-2) \\
\text{autoregressive moving average (ARMA):} &\quad y(t) = a_0 u(t) + a_1 u(t-1) + b_1 y(t-1)
\end{aligned}$$

Note that in the AR and ARMA models, $y(t)$ depends indirectly on all previous inputs, $u(s)$ for $s < t$, due to the recursive dependence on $y(t-1)$. For this reason, the AR and ARMA models are said to have *infinite memory*. The MA model, on the other hand, has a *finite memory*: $y(t)$ depends only on the current and two previous inputs. (Another term for this MA model is 3-tap system, where taps refer to taps on a delay line.)

Each of these models is specified by its parameters, *i.e.*, the scalars c_i, a_i, b_i. For each of these models, find the least-squares fit to the given data. In other words, find parameter values that minimize the sum-of-squares of the residuals. For example, for the ARMA model, pick a_0, a_1, and b_1 that minimize

$$\sum_{t=2}^{N} \left(y(t) - a_0 u(t) - a_1 u(t-1) - b_1 y(t-1)\right)^2 .$$

(Note that we start the sum at $t = 2$ which ensures that $u(t-1)$ and $y(t-1)$ are defined.) For each model, give the root-mean-square (RMS) residual, *i.e.*, the squareroot of the mean of the optimal residual squared. Plot the output $\hat{y}$ predicted by your model, and plot the residual (which is $y - \hat{y}$). The data for this problem are available from the class web page in the file `uy_data.json`. This file contains the vectors u and y and the scalar N (the length of the vectors). Now you can plot u, y, etc. *Note:* the dataset u, y is *not* generated by any of the models above. It is generated by a nonlinear recursion, which has infinite memory.

3. Identifying a system from input/output data. We consider the standard setup:

$$y = Ax + v,$$

where $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$ is the input vector, $y \in \mathbb{R}^m$ is the output vector, and $v \in \mathbb{R}^m$ is the noise or disturbance. We consider here the problem of estimating the matrix A, given some input/output data. Specifically, we are given the following:

$$x^{(1)}, \ldots, x^{(N)} \in \mathbb{R}^n, \qquad y^{(1)}, \ldots, y^{(N)} \in \mathbb{R}^m.$$

These represent N samples or observations of the input and output, respectively, possibly corrupted by noise. In other words, we have

$$y^{(k)} = Ax^{(k)} + v^{(k)}, \quad k = 1, \ldots, N,$$

where $v^{(k)}$ are assumed to be small. The problem is to estimate the (coefficients of the) matrix A, based on the given input/output data. You will use a least-squares criterion to form an estimate $\hat{A}$ of A. Specifically, you will choose as your estimate $\hat{A}$ the matrix that minimizes the quantity

$$J = \sum_{k=1}^{N} \|Ax^{(k)} - y^{(k)}\|^2$$

over A.

a) Explain how to do this. If you need to make an assumption about the input/output data to make your method work, state it clearly. You may want to use the matrices $X \in \mathbb{R}^{n\times N}$ and $Y \in \mathbb{R}^{m\times N}$ given by

$$X = \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \end{bmatrix}, \qquad Y = \begin{bmatrix} y^{(1)} & \cdots & y^{(N)} \end{bmatrix}$$

in your solution.

b) On the course web site you will find some input/output data for an instance of this problem in the file `sysid_data.json`. Executing this Julia file will assign values to m, n, and N, and create two matrices that contain the input and output data, respectively. The $n \times N$ matrix variable `X` contains the input data $x^{(1)}, \ldots, x^{(N)}$ (*i.e.*, the first column of `X` contains $x^{(1)}$, etc.). Similarly, the $m \times N$ matrix `Y` contains the output data $y^{(1)}, \ldots, y^{(N)}$. You must give your final estimate $\hat{A}$, your source code, and also give an explanation of what you did.

4. Curve-smoothing. We are given a function $F : [0, 1] \to \mathbb{R}$ (whose graph gives a curve in $\mathbb{R}^2$). Our goal is to find another function $G : [0, 1] \to \mathbb{R}$, which is a *smoothed* version of F. We'll judge the smoothed version G of F in two ways:

- *Mean-square deviation from F,* defined as

$$D = \int_0^1 (F(t) - G(t))^2 \, dt.$$

- *Mean-square curvature,* defined as

$$C = \int_0^1 G''(t)^2 \, dt.$$

We want *both* D and C to be small, so we have a problem with two objectives. In general there will be a trade-off between the two objectives. At one extreme, we can choose $G = F$, which makes $D = 0$; at the other extreme, we can choose G to be an affine function (*i.e.*, to have $G''(t) = 0$ for all $t \in [0, 1]$), in which case $C = 0$. The problem is to identify the optimal trade-off curve between C and D, and explain how to find smoothed functions G on the optimal trade-off curve. To reduce the problem to a finite-dimensional one, we will represent the functions F and G (approximately) by vectors f, $g \in \mathbb{R}^n$, where

$$f_i = F(i/n), \quad g_i = G(i/n).$$

You can assume that n is chosen large enough to represent the functions well. Using this representation we will use the following objectives, which approximate the ones defined for the functions above:

- *Mean-square deviation,* defined as

$$d = \frac{1}{n} \sum_{i=1}^{n} (f_i - g_i)^2.$$

- *Mean-square curvature,* defined as

$$c = \frac{1}{n-2} \sum_{i=2}^{n-1} \left(\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2} \right)^2.$$

In our definition of c, note that

$$\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2}$$

gives a simple approximation of $G''(i/n)$. You will only work with this approximate version of the problem, *i.e.*, the vectors f and g and the objectives c and d.

a) Explain how to find g that minimizes $d + \mu c$, where $\mu \geq 0$ is a parameter that gives the relative weighting of sum-square curvature compared to sum-square deviation. Does your method always work? If there are some assumptions you need to make (say, on rank of some matrix, independence of some vectors, etc.), state them clearly. Explain how to obtain the two extreme cases: $\mu = 0$, which corresponds to minimizing d without regard for c, and also the solution obtained as $\mu \to \infty$ (*i.e.*, as we put more and more weight on minimizing curvature).

b) Get the file `curve_smoothing.json` from the course web site. This file defines a specific vector f that you will use. Find and plot the optimal trade-off curve between d and c. Be sure to identify any critical points (such as, for example, any intersection of the curve with an axis). Plot the optimal g for the two extreme cases $\mu = 0$ and $\mu \to \infty$, and for three values of μ in between (chosen to show the trade-off nicely). On your plots of g, be sure to include also a plot of f, say with dotted line type, for reference. Submit your matlab code.

5. Hovercraft with limited range. We have a hovercraft moving in the plane with two thrusters, each pointing through the center of mass, exerting forces in the **x** and **y** directions with 100% efficiency. The hovercraft has mass 1. The discretized equations of motion for the hovercraft are

$$x(t+1) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & 0 \\ 0 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

where x_1 and x_2 are the position and velocity in the **x**-direction, and x_3, x_4 are the position and velocity in the **y**-direction. Here

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

is the force acting on the hovercraft for time in the interval $[t, t+1)$. Let the position of the vehicle at time t be $q(t) \in \mathbb{R}^2$.

a) The hovercraft starts at the origin. We'd like to apply thrust to make it move through points p_1, p_2, p_3 at times t_1, t_2, t_3, where

$$p_1 = \begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix} \qquad p_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad p_3 = \begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}$$

$$t_1 = 6 \qquad t_2 = 40 \qquad t_3 = 50$$

We will run the hovercraft on the time interval $[0, 70]$. We'd like to apply a sequence of inputs $u(0), u(1), \ldots, u(70)$ to make the hovercraft position pass through the above sequence of points at the specified times.

We would like to find the sequence of inputs that drives the hovercraft through the desired points which has the minimum cost, given by the sum of the squares of the forces:

$$\sum_{t=0}^{70} \|u(t)\|^2$$

To do this, pick A_{hov} and y_{des} to set this problem up as an equivalent minimum-norm problem, where we would like to find the minimum-norm u_{seq} which satisfies

$$A_{\text{hov}} u_{\text{seq}} = y_{\text{des}}$$

where u_{seq} is the sequence of force inputs

$$u_{\text{seq}} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(70) \end{bmatrix}$$

Plot the trajectory of the hovercraft using this input, and the way-points $p_1, \ldots, p_3$. Also plot the optimal u against time.

b) Now we would like to compute the trade-off curve between the accuracy with which the mass passes through the waypoints and the norm of the force used. Let our two objective functions be

$$J_1 = \sum_{i=1}^{3} \|q(t_i) - p_i\|^2 = \|A_{\text{hov}} u_{\text{seq}} - y_{\text{des}}\|^2$$

and

$$J_2 = \sum_{t=0}^{70} \|u(t)\|^2$$

By minimizing the weighted sum

$$J_1 + \mu J_2$$

for a range of values of μ, plot the trade-off curve of J_1 against J_2 showing the achievable performance. To generate suitable values of μ, you may find the `logspace` command useful in Matlab; you'll need to pick appropriate maximum and minimum values. This above trade-off curve shows how we can trade-off between how accurately the hovercraft passes through the waypoints and how much input energy is used.

c) For each of the following values of μ

$$\{\, 10^{\frac{p}{2}} \mid p = -2, 0, 2, \ldots, 10 \,\}$$

plot the trajectories all on the same plot, together with the waypoints.

d) Now suppose we are controlling the hovercraft by radio control, and the maximum range possible between the transmitter and receiver is 2 (in whatever units we are using for distance.) Notice that, if we use the minimum-norm input then the hovercraft passes out of range, both when making its first turn and on the final stretch (between times 50 and 70).

We'd like to do something about this, but trading off the input norm as above doesn't do the right thing; if μ is large then the hovercraft stays within range, but misses the waypoints entirely; if μ is small then it comes close to the waypoints, but goes out of range. Notice that this is particularly a problem on the final stretch between times 50 and 70; explain why this is.

e) One remedy for this problem is to solve a *constrained multiobjective least-squares* problem. We would like to impose the constraint that

$$A_{\text{hov}} u_{\text{seq}} = y_{\text{des}}$$

that is, achieve zero waypoint error $J_1 = 0$. We can attempt to keep the hovercraft in range by trading off the sum of the squares of the *position*

$$J_3 = \sum_{t=0}^{70} \|q(t)\|^2$$

against input cost J_2 subject to this constraint. To do this, we'll solve

$$\begin{array}{ll} \text{minimize} & J_3 + \gamma J_2 \\ \text{subject to} & A_{\text{hov}} u_{\text{seq}} = y_{\text{des}} \end{array}$$

First, find the matrix W so that the cost function is given by

$$J_3 + \gamma J_2 = \|W u_{\text{seq}}\|^2$$

f) Now we have a problem of the form

$$\begin{array}{ll} \text{minimize} & \|Wu\|^2 \\ \text{subject to} & Au = y_{\text{des}} \end{array}$$

This is called a *weighted minimum-norm solution*; the only difference from the usual minimum-norm solution to $Au = y_{\text{des}}$ is the presence of the matrix W, and when $W = I$ the optimal u is just given by $u_{\text{opt}} = A^\dagger y_{\text{des}}$. Show that the solution for general W is

$$u_{\text{opt}} = \Sigma^{-1} A^T (A \Sigma^{-1} A^T)^{-1} y_{\text{des}}$$

where $\Sigma = W^T W$. (One way to do this is using Lagrange multipliers.) Use this to solve the remaining parts of this problem.

g) For each of the following values of γ

$$\{\, 10^{\frac{p}{2}} \mid p = 0, 2, 4, \ldots, 20 \,\}$$

Plot the trajectories all on the same plot, together with the waypoints. Explain what you see.

h) By trying different values of γ, you should be able to find a trajectory which just keeps the hovercraft within range. Plot the trajectory of the hovercraft; what is the corresponding value of γ? Is this the smallest-norm input u that just keeps the hovercraft within range, and drives the hovercraft through the waypoints? Explain why, or why not.

i) For a range of values of γ, plot the trade-off curve of J_3 against J_2 showing the achievable performance.

6. You Must Construct Additional Pylons. You are the Hierarch of the Baelaam charged with maintaining the power levels of energizing pylons which power various structures in your base of operations. Consider m structures powered by n pylons. Each structure's energy level y_j for $j = 1...m$ is given by

$$y_j(p) = log(\sum_{i=1}^{n} \exp(\frac{p_i}{d_{j,i}^2}))$$

Where p_i are the power levels of the i'th pylon and $d_{j,i}$ are the distances between the j'th structure and the i'th pylon (we choose log-sum-exp as a smooth approximation of the max function). While each structure has some given target energy level R_j, they can handle some deviation (either over or under), however that will cause damage to the Nexus Crystals that act as energy conduits for the structure. Your goal as Hierarch is to find a set of pylon power levels $\mathbf{p} \in \mathbb{R}^n$ that minimizes the total square deviation, J, from the required energy levels.

$$J(p) = \sum_{j=1}^{m} (R_j - y_j(p))^2$$

Your chief engineer proposes that you could linearize the $y_j(p)$ function to find an update algorithm that starts with some initial pylon power level and changes the power each step by a small amount to reduce the total energy deviation J.

a) Find an update expression for the approximate power level $\mathbf{y}(p+\delta p)$ as a linear dynamical system where $\mathbf{y} \in \mathbb{R}^m$ is the vector of structure energy levels. I.E find A and B such that

$$\mathbf{y}(p + \delta p) \approx A\mathbf{y}(p) + B\delta p$$

We want to relate the energy level at $p + \delta p$ to the energy level at p and the change in energy from a small change in power δp.

hint: B is not necessarily constant

b) Derive an expression for the one step change in power levels that minimizes

$$J(p + \delta p) = \sum_{j=1}^{m} (R_j - y_j(p + \delta p))^2$$

as a function of $y(p), A, B, \delta p$. Use the result of this minimization problem, (the optimal δp) to determine an update expression for $p[k+1] = p[k] + \alpha\delta p$, where α is a given step size, and k is the current iteration. If your method involves an inverse, explain what conditions must hold in order for the inverse to exist.

c) Given the following list of required energy levels and locations of each structure and pylon, apply your algorithm for 200 iterations with an $\alpha = .01$ and in initial power level of $p[0]$ =[20, 40, 20].

Plot the pylon power levels and the structures energy levels for each iteration. as well as the the power deviation metric J. There should be 3 plots total. It should converge in roughly 150-200 iterations.

Also report the final cost and pylon power levels

```
Stucture_energy_goal=[10, 20 , 5 , 10 , 5]
Strcuture_location=[2 8; 4 5; 6 8; 2 2; 4 1]
Pylon_location=[2 5; 3 4 ; 5 4]
p0=[20 40 20]
```

For locations, each row is an x,y location.