Homework 7: Convex programs

Due date: 11:59pm on Friday 26 March 2021
See the course website for instructions and submission details.

1. Moving averages. There are many ways to model the relationship between an input sequence $\{u_1, u_2, \dots\}$ and an output sequence $\{y_1, y_2, \dots\}$. In class, we saw the *moving average* (MA) model, where each output is approximated by a linear combination of the k most recent inputs:

$$\text{MA:} \qquad y_t \approx b_1 u_t + b_2 u_{t-1} + \cdots + b_k u_{t-k+1}$$

We then used least-squares to find the coefficients $b_1, \dots, b_k$. What if we didn't have access to the inputs at all, and we were asked to predict future y values based *only* on the previous y values? One way to do this is by using an *autoregressive* (AR) model, where each output is approximated by a linear combination of the ℓ most recent outputs (excluding the present one):

$$\text{AR:} \qquad y_t \approx a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_\ell y_{t-\ell}$$

Of course, if the inputs contain pertinent information, we shouldn't expect the AR method to outperform the MA method!

a) Using the same dataset from class `uy_data.csv`, plot the true y, and on the same axes, also plot the estimated $\hat{y}$ using the MA model and the estimated $\hat{y}$ using the AR model. Use $k = 5$ for both models. To quantify the difference between estimates, also compute $\|y - \hat{y}\|$ for both cases.

b) Yet another possible modeling choice is to combine both AR and MA. Unsurprisingly, this is called the *autoregressive moving average* (ARMA) model:

$$\text{ARMA:} \qquad y_t \approx a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_\ell y_{t-\ell} + b_1 u_t + b_2 u_{t-1} + \cdots + b_k u_{t-k+1}$$

Solve the problem once more, this time using an ARMA model with $k = \ell = 1$. Plot y and $\hat{y}$ as before, and also compute the error $\|y - \hat{y}\|$.

Note: For the problems in this question you don't need to use optimization codes; you can just use the "backslash" notation for solving linear least squares.

2. The Huber loss. In statistics, we frequently encounter data sets containing *outliers*, which are bad data points arising from experimental error or abnormally high noise. Consider for example the following data set consisting of 15 pairs (x, y).

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y	6.31	3.78	24	1.71	2.99	4.53	2.11	3.88	4.67	4.25	2.06	23	1.58	2.17	0.02

The y values corresponding to $x = 3$ and $x = 12$ are outliers because they are far outside the expected range of values for the experiment.

a) Compute the best linear fit to the data using an ℓ_2 cost (least squares). In other words, we are looking for the a and b that minimize the expression:

$$\ell_2 \text{ cost:} \qquad \sum_{i=1}^{15} (y_i - a x_i - b)^2$$

Repeat the linear fit computation but this time exclude the outliers from your data set. On a single plot, show the data points and both linear fits. Explain the difference between both fits.

b) It's not always practical to remove outliers from the data manually, so we'll investigate ways of automatically dealing with outliers by changing our cost function. Find the best linear fit again (including the outliers), but this time use the ℓ_1 cost function:

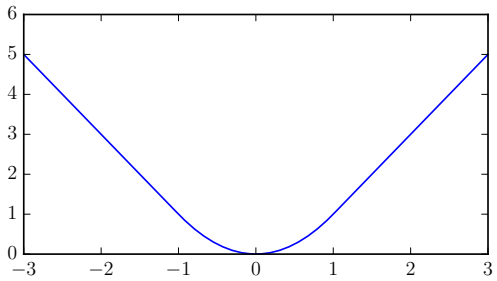
$$\ell_1 \text{ cost:} \qquad \sum_{i=1}^{15} |\, y_i - ax_i - b \,|$$

Include a plot containing the data and the best ℓ_1 linear fit. Does the ℓ_1 cost handle outliers better or worse than least squares? Explain why.

c) Another approach is to use an ℓ_2 penalty for points that are close to the line but an ℓ_1 penalty for points that are far away. Specifically, we'll use something called the *Huber loss*, defined as:

$$\phi(x) = \begin{cases} x^2 & \text{if } -M \le x \le M \\ 2M|x| - M^2 & \text{otherwise} \end{cases}$$

Here, M is a parameter that determines where the quadratic function transitions to a linear function. The plot on the right shows what the Huber loss function looks like for $M = 1$.



The formula above is simple, but not in a form that is useful for us. As it turns out, we can evaluate the Huber loss function at any point x by solving the following convex QP instead:

$$\phi(x) = \left\{ \begin{array}{ll} \underset{v,w}{\text{minimize}} & w^2 + 2Mv \\ \text{subject to:} & |x| \le w + v \\ & v \ge 0,\ w \le M \end{array} \right\}$$

Verify this fact by solving the above QP (with $M = 1$) for many values of x in the interval $-3 \le x \le 3$ and reproducing the plot above. Finally, find the best linear fit to our data using a Huber loss with $M = 1$ and produce a plot showing your fit. The cost function is:

$$\text{Huber loss:} \qquad \sum_{i=1}^{15} \phi(\, y_i - ax_i - b)$$

3. Heat pipe design. A heated fluid at temperature T (degrees above ambient temperature) flows in a pipe with fixed length and circular cross section with radius r. A layer of insulation, with thickness w, surrounds the pipe to reduce heat loss through the pipe walls (w is much smaller than r). The design variables in this problem are T, r, and w.

The energy cost due to heat loss is roughly equal to $\alpha_1 Tr/w$. The cost of the pipe, which has a fixed wall thickness, is approximately proportional to the total material, i.e., it is given by $\alpha_2 r$. The cost of the insulation is also approximately proportional to the total insulation material, i.e., roughly $\alpha_3 rw$. The total cost is the sum of these three costs.

The heat flow down the pipe is entirely due to the flow of the fluid, which has a fixed velocity, i.e., it is given by $\alpha_4 Tr^2$. The constants α_i are all positive, as are the variables T, r, and w.

Now the problem: maximize the total heat flow down the pipe, subject to an upper limit $C_{\max}$ on total cost, and the constraints

$$T_{\min} \le T \le T_{\max}, \qquad r_{\min} \le r \le r_{\max} \qquad w_{\min} \le w \le w_{\max}, \qquad w \le 0.1r$$

a) Express this problem as a geometric program, and convert it into a convex optimization problem.

b) Consider a simple instance of this problem, where $C_{\max} = 500$ and $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$. Also assume for simplicity that each variable has a lower bound of zero and no upper bound. Solve this problem using JuMP. Use the `Ipopt` solver and the command `@NLconstraint(...)` to specify nonlinear constraints such as log-sum-exp functions. Have your code print the optimal values of T, r, and w, as well as the optimal objective value.