Homework 7: Quadratic Programming

Due date: Friday April 8, 2022
See the course website for instructions and submission details.

1. **Quadratic form positivity.** You're presented with the constraint:

$$2x^2 + 2y^2 + 9z^2 + 8xy - 6xz - 6yz \le 1 \tag{1}$$

a) Write the constraint (1) in the standard form $v^\mathsf{T} Q v \le 1$. Where Q is a symmetric matrix. What is Q and what is v?

b) It turns out the above constraint is *not* convex. In other words, the set of (x, y, z) satisfying the constraint (1) is not an ellipsoid. Explain why this is the case.

Note: you can perform an orthogonal decomposition of a symmetric matrix Q in Julia like this:

```
(L,U) = eigen(Q)    # L is the vector of eigenvalues and U is orthogonal
U * diagm(L) * U'   # this is equal to Q (as long as Q was symmetric to begin with)
```

c) We can also write the constraint (1) using norms by putting it in the form:

$$\|Av\|^2 - \|Bv\|^2 \le 1$$

What is v and what are the matrices A and B that make the constraint above equivalent to (1)?

d) Explain how to find a direction for vector (x, y, z) such that $\|(x, y, z)\|_2$ (norm) is unbounded (can be made arbitrarily large) while satisfying the constraint (1).

2. **Enclosing circle.** Given a set of points in the plane $x_i \in \mathbb{R}^2$, we would like to find the circle with smallest possible area that contains all of the points. Explain how to model this as an optimization problem. To test your model, generate a set of 50 random points using the code `X = 4.+randn(2,50)` (this generates a 2×50 matrix X whose columns are the x_i). Produce a plot of the randomly generated points along with the enclosing circle of smallest area.

To get you started, the following Julia code generates the points and plots a circle:

```
using PyPlot
X = 4 .+ randn(2,50)                          # generate 50 random points
t = range(0,stop=2pi,length=100)                     # parameter that traverses the circle
r = 2; x1 = 4; x2 = 4                        # radius and coordinates of the center
plot( x1 .+ r*cos.(t), x2 .+ r*sin.(t))       # plot circle radius r with center (x1,x2)
scatter( X[1,:], X[2,:], color="black")      # plot the 50 points
axis("equal")                                # make x and y scales equal
```

3. **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	5.12	1.71	2.99	4.53	2.11	3.88	4.67	26	2.06	23	1.58	2.17	0.02

The y values corresponding to $x = 10$ 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:

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

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:

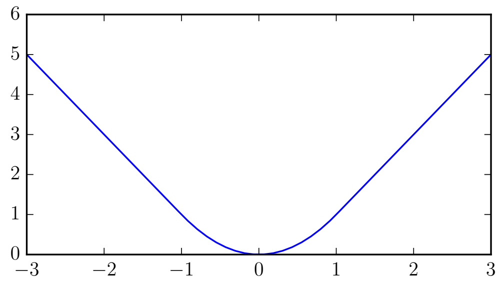
$$\ell_1 \text{ cost:} \qquad \sum_{i=1}^{15} |\, y_i - ax_i - b\,|$$

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:

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