ORF 523	Problem set 6	Spring 2024, Princeton University
Instructor: A.A. Ahmadi		
AIs: A.Z. Chaudhry, Y. Hua		
Due on May 2, 2024, at 1:30pm EST, on Gradescope		

Problem 1: Equivalence of decision and search for some problems in NP

1. Suppose you had a blackbox that given a 3SAT instance would tell you whether it is satisfiable or not. How can you make polynomially many calls to this blackbox to find a satisfying assignment to any satisfiable instance of 3SAT?

2. Suppose you had a blackbox that given a graph G and an integer k would tell you whether G has a stable set of size larger or equal to k. How can you make polynomially many calls to this blackbox to find a maximum stable set of a given graph?

Problem 2: Complexity of rank-constrained SDPs

Consider a family of decision problems indexed by a positive integer k:

RANK-k-SDP
Input: Symmetric $n \times n$ matrices $A_1, \ldots, A_m$ with entries in $\mathbb{Q}$, scalars $b_1, \ldots, b_m \in \mathbb{Q}$.
Question: Is there a real symmetric matrix X that satisfies the constraints

$$\mathrm{Tr}(A_i X) = b_i, i = 1, \ldots, m,\ X \succeq 0,\ \mathrm{rank}(X) = k?$$

Show that RANK-k-SDP is NP-hard for any integer $k \geq 1$.

(Hint: First show NP-hardness for $k = 1$, then see how you can modify your construction so that it would work for any other k.)

Problem 3: Complexity of testing monotonicity

A polynomial $p(x) := p(x_1, \ldots, x_n)$ is nondecreasing with respect to a variable x_i if

$$\frac{\partial p}{\partial x_i}(x) \geq 0, \forall x \in \mathbb{R}^n.$$

Show that the problem of deciding whether a degree-d polynomial[1] with rational coefficients is nondecreasing with respect to a particular variable (e.g., x_1) is

[1]Here, the degree of a polynomial is equal to the highest degree of its monomials with a nonzero coefficient.

(i) in P if d is less than 5,[2]

(ii) NP-hard if d is greater than or equal to 5.

Problem 4: Monotone and convex regression

In the previous problem, we saw that deciding whether a polynomial is monotone is NP-hard. The same claim holds for checking convexity of polynomials (of degree $2d \geq 4$). This suggests that optimizing over monotone or convex polynomials will naturally also be NP-hard. Nonetheless, in this problem, we explore some ways to perform this task.

1. In the file `regression_data.mat`, you are given 20 points (x_i, f_i) in $\mathbb{R}^2$ where $(x_i)_{i=1,\ldots,20}$ are the entries of the vector `xvec` and $(f_i)_{i=1,\ldots,20}$ are the entries of the vector `fvec`.

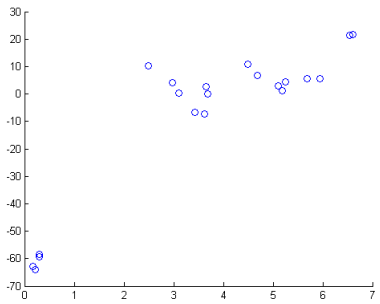

Figure 1: Figure generated by `scatter(xvec,fvec)`

The goal is to fit a polynomial of degree 7

$$p(x) = c_0 + c_1 x + \ldots c_7 x^7 \tag{1}$$

to the data to minimize least square error:

$$\min_{c_0,c_1,\ldots,c_7} \sum_{i=1}^{20} (p(x_i) - f_i)^2. \tag{2}$$

The data comes from noisy measurements of an unknown function that is a priori known to be nondecreasing (e.g., the number of calories you intake as a function of the number of Big Macs you eat).

[2]You can take as given that positive semidefiniteness of an $r \times r$ matrix can be checked in time $O(r^3)$.

(a) If the underlying function is truly monotone and the noise is not too large, one may hope that least squares would automatically respect the monotonicity constraint. Solve (2) to see if this is the case. Plot the optimal polynomial you get and report the optimal value.

(b) Resolve (2) subject to the constraint that the polynomial (1) is nondecreasing. Plot the optimal polynomial you get and report the optimal value.

2. In the file `regression_data.mat`, you are also given 30 points (x_i^1, x_i^2, g_i) in $\mathbb{R}^3$ where $(x_i^1)_{i=1,\ldots,30}$ are the entries of the vector `x1vec`, $(x_i^2)_{i=1,\ldots,30}$ are the entries of `x2vec` and $(g_i)_{i=1,\ldots,30}$ are the entries of the vector `gvec`. You can see these points below.

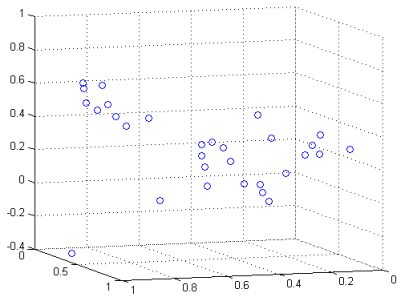

Figure 2: Figure generated by `scatter3(x1vec,x2vec,gvec)`

The goal in this case is to fit a polynomial of degree 4

$$p := p(x_1, x_2) = c_0 + c_1x_1 + c_2x_2 + c_3x_1^2 + c_4x_1x_2 + c_5x_2^2 + \ldots c_{15}x_2^4$$

to the data to minimize least square error:

$$\min_{c_0,c_1,\ldots,c_{15}} \sum_{i=1}^{30} (p(x_i) - g_i)^2. \tag{3}$$

This time, the unknown underlying function is known to be convex; we want this property to be preserved in our regression.

(a) Solve (3) and plot the resulting polynomial together with the data points. Report the optimal value of the problem (denoted by η^*). Is the optimal polynomial convex?

(b) Find a convex polynomial p of degree no more than 4 such that its least squares error

$$\eta := \sum_{i=1}^{30} (p(x_i) - g_i)^2$$

satisfies $\eta < 1.75\eta^*$.

Coding instructions

- YALMIP is the recommended SOS parser in MATLAB. Take advantage of the built-in functions of YALMIP such as `hessian, jacobian`, etc.
- In Python, you could install sympy and the SumOfSquares package. Use the functions `SOSproblem(), set_objective(), add_sos_constraint()` in the SumOfSquares package to create an SOS problem, and use the `diff` function in sympy to compute the gradient and the Hessian.