Quadratic programming(?) for fun and profit
m_powers 1d

I've been working off and on with a constrained optimization problem for (auction-style) fantasy football. The idea is to fill each of N roster spots (where N is usually between 5 and 9) with options from a pool of NFL players. Each player in the pool has a projected points outcome, and will cost a certain salary to claim. The objective is to maximize the sum of points $P = \Sigma_{p_n}$ from the N player selections, subject to the constraint that the sum of salaries $S = \Sigma_{s_n}$ must be less than a given auction cap C.

In the past I've approached this as a sort of knapsack problem which would fall under the combinatorial optimization category described here. But I've realized that I can construct "cost curves" relating salary and points, and these curves are differentiable, which I think could place this problem into something closer to categories discussed in this class.

Specifically, for each roster spot R_n I have a cost curve of the form relating salary to points that looks something like this: $s_n = a_n p_n^2 + b_n p_n + c_n$. In other words, the more points a player is expected to produce, the cost gets higher in nonlinear (quadratic) fashion. Solving the relation in reverse, that looks more like $p_n = sqrt(-4a_n c_n + 4a_n s_n + b_n^2)/2a_n$ (These cost curves have different, known (a_n, b_n, c_n) for each roster spot; technically various roster spots draw from different subpools of players). Also, there are no "interaction" terms between the s_n other than through the constraint on S.

So I'm looking for a solution vector of scalar salaries $[s_0 \ldots s_n]$ that maximizes P subject to $S <= C$. (The individual s_n should also be subject to min and max constraints if there is a way to accommodate that).

Is this even quadratic programming? The "cost curves" are in quadratic form in terms of the input variables s_n, but not in terms of the objective P to be maximized. Either way, any thoughts or suggestions for how to set this up in a form similar to ones we are discussing in class?

Related to 13.2 Multi-variable Optimization / 13.2.5 Types of Optimization Problems

Showing 2 responses Newest first

sandipan_dey 6h

The problem is analytically solvable.

We have the constrained optimization problem:

$$\max_{s_1,s_2,\ldots,s_n} \sum_n \frac{-b_n+\sqrt{b_n^2-4a_n(c_n-s_n)}}{2a_n} = \max_{s_1,s_2,\ldots,s_n} \sum_n \sqrt{K+s_n}, \quad s.t.,\sum_n s_n \le C,$$

where $K = \frac{b^2}{4a_n} - c_n$, a constant.

The problem is not quadratic since the objective function is not.

The Lagrange multiplier to convert the constrained optimization problem into an unconstrained one:

$$L(s_1, s_2, \ldots, s_n, \lambda) = \sum_n \sqrt{K+s_n} + \lambda\left(\sum_n s_n - C\right)$$

At a critical point, we have,

$$\frac{\partial L}{\partial s_i} = \frac{1}{2\sqrt{K+s_i}} + \lambda = 0 \quad \forall i$$

$$\frac{\partial L}{\partial \lambda} = \sum_n s_n - C = 0$$

which leads to the solution $s_1 = s_2 = \ldots = s_n = \frac{C}{n}$

We can compute the Hessian and observe that it's negative semidefinite (since it will only have nonzero negative elements on the principal diagonal) and hence existence of a local maximum is confirmed at the point.

sandipan_dey 3m •••
@m_powers Ok then the problem becomes the following:

$$\max_{s_1,s_2,\ldots,s_n} \sum_n \frac{-b_n+\sqrt{b_n^2-4a_n(c_n-s_n)}}{2a_n} = \max_{s_1,s_2,\ldots,s_n} \sum_n \sqrt{K_n+s_n}, \quad s.t.,\sum_n s_n \le C,$$

where $K_n = \frac{b^2}{4a_n} - c_n$, we have n constants

The Lagrange multiplier to convert the constrained optimization problem into an unconstrained one

$$L(s_1, s_2, \ldots, s_n, \lambda) = \sum_n \sqrt{K_n+s_n} + \lambda\left(\sum_n s_n - C\right)$$

At a critical point, we have,

$$\frac{\partial L}{\partial s_i} = \frac{1}{2\sqrt{K_i+s_i}} + \lambda = 0 \quad \forall i$$

$$\frac{\partial L}{\partial \lambda} = \sum_n s_n - C = 0$$

$$\implies K_1 + s_1 = K_2 + s_2 = \ldots = K_n + s_n$$

$$\implies s_i = K_1 - K_i + s_1, \ \forall i \ge 2$$

$$\implies \sum_n s_n = s_1 + s_2 + \ldots + s_n = nK_1 - \sum_n K_n + ns_1 = C$$

The following figure shows solution for 2D:

5 1d

Visualizing additional dimensions? No preview available
m_powers
4 1d

PSET 3: Assertion Errors in Contour Plot For PSET 3 contou ...
sandipan_dey
4 2d

step size times J-prime No preview available
m_powers
4 2d

Stiffness issue? No preview available
JavierM0401
3 1d

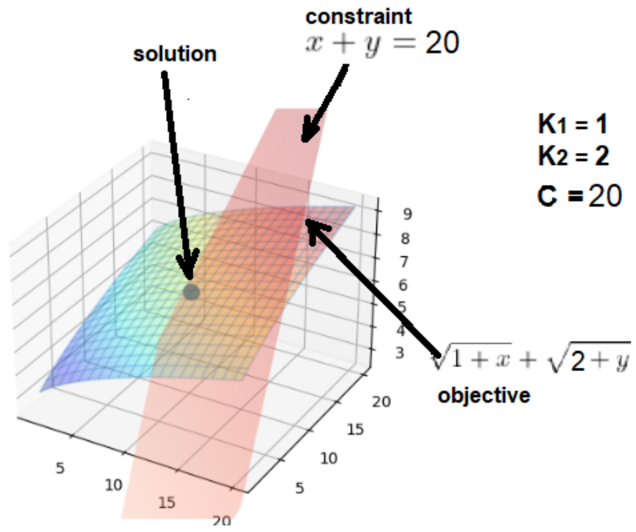
inconsistent plotting label why is it that in the loop, the (XX) is ...
pkchong79
3 1d

Exam 1 code Hi, will solutions for the coding problems in Exam 1 ...
dyaeger
2 1d

Solutions to the problem sets Are the solutions to the problem s ...
Vasiliki_Ts
2 1d

Eq. 4.50 Where does the exponent of -1 on the square brack ...
stp2004
2 1d

Load more posts



m_powers 4h

Thanks for this analysis @sandipan_dey! I don't quite follow it all yet, but I believe that there is a mistake somewhere, perhaps in my description, because $s_1 = s_2 = \ldots = s_n = \frac{C}{n}$ is not in my experience a general solution to the real-life version of the problem that I am trying to describe.

In particular, the "cost curve" (points vs. salary) is different for each roster spot because those spots draw from different subpools (football positions). I was trying to convey this by saying that the constant coefficients a_n, b_n, c_n have different values for each of the equations in the system. So you are correct that K is a constant as you have formulated it, but there would be a different K_n for each of the equations in the system.

Intuitively, if your QB-position roster spot has a "steep" salary response to points, and the RB-position roster spot has a "shallow" salary response to points, placing more salary into the RB spot than the QB spot will yield a greater points total. So I would expect the s_n to be different based on the particulars of the different cost curves.

Add comment

darmofal Staff 10h

So, there is such a think as "quadratic programming with quadratic constraints" and some methods designed to solve it. In general, there are no guarantees on being able to solve it, unless your particular problem has certain properties.

In terms of the gradient descent approach we have been looking at, the main issue is how to add constraints. The most common approach for doing this is to add a penalty term which gets large as the constraints are violated. So, your objective function might look like:

$$J = P + f(C - S)$$

where $f(x)$ is a function that is smooth, and for $x < 0$ is designed to have $f(x) \approx 0$ but increases rapidly as $x > 0$. This is not quite the constrained optimization problem, however the method is very commonly used to approximately solve constrained optimization problems. A related approach is to introduce the constraints using so-called Lagrange multiplier. Unfortunately, constrained optimization problems are outside the scope of this class (yes, I say that often). But there are many places to learn about such algorithms.

Great question!

Add a response