Convex Optimisation Problems

COMP4691-8691
School of Computer Science, ANU

1 Convex Functions

For a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, state a condition that f needs to satisfy in order to be a convex function if f is not differentiable. State a different condition for the case where f *is* differentiable.

2 Is a Convex Function?

Prove whether or not $f(x, y) := x^2 + xy - 2$ is convex.

3 Penalty Method

Explain the penalty method, using the following all-inequality constrained problem as an example:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & f(x) \\ \text{s.t. } & g(x) \leq 0 \end{aligned}$$

4 Dual Problem

Write out the dual problem, in terms of the Lagrangian $\mathcal{L}(x, \mu, \lambda)$ for a minimisation problem. What is the relationship between the solution of the dual and primal problem? Briefly explain the steps of the dual gradient ascent algorithm.

5 KKT

Show that the KKT conditions are satisfied for the following problem when $(x, y) = (1.5811, 0.6325)$. What are the implications of this?

$$\begin{aligned} \min_{x,y} \ & (y+2)^2 \\ \text{s.t. } & y \geq \frac{1}{x} \\ & 2x^2 \leq 5 \\ & x \geq 1 \end{aligned}$$

6 Convex Relaxation

Take a convex relaxation of the following set of constraints (need not be linear):

$$\begin{aligned} & y = \cos(\theta) \\ & \theta \in [-\frac{\pi}{2}, \frac{\pi}{2}] \end{aligned}$$

7 Method Order

Discuss the tradeoffs between first and second order methods.

8 Dual Gradient Ascent

Dual gradient ascent is an algorithm for solving strictly convex constrained optimisation problems. In this question you will implement the approach, and use it to solve several problems.

The `problems.py` file contains two constrained optimisation problems hard-coded into classes `Problem1` and `Problem2`. These classes offer functions to evaluate the objective, the gradient of the objective, the constraints and the constraint jacobian. This file also contains a more general class to represent quadratically constrained quadratic programs (QCQP). It can load a specific problem instance from file.

Your code should go in the file `dual_ascent.py`. This can be called as a script. If you provide it with a JSON file it will solve the corresponding QCQP, otherwise it will solve the two hardcoded `Problem1` and `Problem2`.

8.1 Gradient Descent

As a subcomponent, we need to be able to first solve gradient descent. Implement the gradient descent algorithm in the `gradient_descent` function, of the `dual_ascent.py` file.

Solve unconstrained versions of `Problem1` and `Problem2` (just ignore the constraints) using your gradient descent algorithm. Report the number of iterations required and the solution.

Hint: You'll likely want to make use of one of the rule-based step sizes, e.g., one of those by Barzilai and Borwein from the lectures.

Hint: Implement a stopping criteria that checks whether the gradient is within a provided tolerance of being zero.

8.2 Dual Gradient Ascent

Next implement the dual ascent algorithm in `dual_ascent`, making use of `gradient_descent`.

Solve the **constrained** versions of `Problem1` and `Problem2` using your algorithm. Report the number of iterations required and the solution.

Hint: Implement a stopping criteria based on satisfaction of the KKT conditions to within a tolerance.

8.3 More General Quadratic Problems

Report the results you get by running your dual ascent algorithm on instances `qcqp-{1,2,3,4}.json`. Based on the dual variables you obtain, explain which constraints are active for each instance. You will likely struggle to solve the instance `qcqp-3.json`. If so explain why.