Lecture 9

Math Foundations Team

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

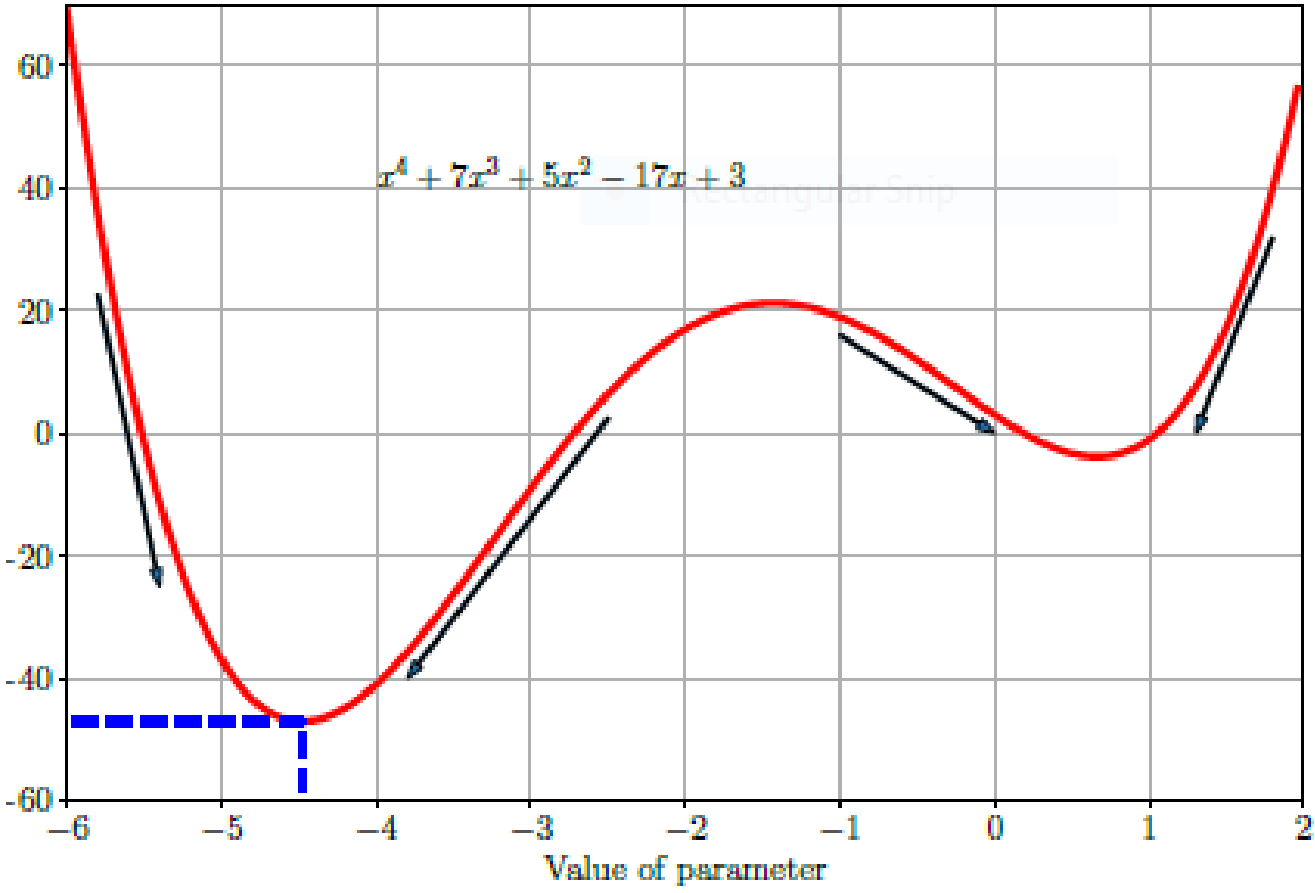
Introduction

- We will look at continuous optimization concepts in this lecture.
- There are two main branches of continuous optimization - constrained and unconstrained optimization.
- We seek the minimum of an objective function which we assume is differentiable.
- This is like finding the valleys of the objective function, and since the objective function is differentiable, the gradient tells us the direction to move to get the maximum increase in the objective function

Unconstrained Optimization

- We move in the direction of the negative gradient to decrease the objective function.
- We move until we encounter a point at which the gradient is zero.
- This constitutes a valley of the objective function, known as a local minimum
- For unconstrained optimization, this is enough for the big picture.

Example


$x^4 + 7x^3 + 5x^2 - 17x + 3$
60
40
20
0
-20
-40
-60
−6
−5
−4
−3
−2
−1
0
1
2
Value of parameter

Example

- Let $I(x) = x^4 + 7x^3 + 5x^2 - 17x + 3$.
- The gradient is $\frac{dI(x)}{dx} = 4x^3 + 21x^2 + 10x - 17$.
- Setting the gradient to zero identifies points corresponding to a local minimum or local maximum - there are three such points since this is a cubic equation.
- How do we check if we are dealing with a local minimum or local maximum? Look at the second derivative!
- If the second derivative at the point where the gradient vanishes is negative we are talking about a local maximum, otherwise it is a local minimum.
- The second derivative is $12x^2 + 42x + 10$

Optimization using gradient descent

- For low-order polynomials we can solve the equations analytically and find points at which the gradient is zero. Then we can do the second derivative test and identify whether these points are local minima/local maxima.
- In general we need to follow the negative gradient.
- Consider the problem of solving for the minimum of a real-valued function $min_{\boldsymbol{x}} f(\boldsymbol{x})$ where $f : R^d \rightarrow R$ is an objective loss function that captures the cost of using the current set of parameters.
- We assume our function f is differentiable but that the minimum cannot be found analytically in closed form.

Optimization using gradient descent

- The main idea of gradient-descent, a first-order optimization algorithm, is to take a step from the current point of magnitude proportional to the negative gradient of the function at the current point.
- In mathematical terms if $\boldsymbol{x}_1 = \boldsymbol{x}_0 - \gamma((\nabla f)(\boldsymbol{x}_0))^T$ for a small step-size $\gamma > 0$ then $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_0)$.
- The above suggests that in order to find the optimum of f, say at $f(\boldsymbol{x}_*)$, we can start at some initial point $\boldsymbol{x}_0$ and then iterate according to $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^T$
- For a suitable step-size γ_i, the sequence of points $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x}_1) \geq \ldots$ converges to some local minimum.

Example of gradient descent

- Let $f = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x}$ where $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix}$ and $\boldsymbol{b} = \begin{bmatrix} -5 \\ -3 \end{bmatrix}$.
- From a previous lecture on Vector Calculus, we know that $\nabla f = \boldsymbol{x}^T\boldsymbol{A} + \boldsymbol{b}^T$
- Starting at $\boldsymbol{x}_0 = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$ and iterating according to the equation given on a previous slide, we will eventually encounter the minimum value.

Choosing step-size

- If the step-size is too small, gradient descent will become too slow.
- If the step-size is too large, gradient descent might overshoot the target, failt to coverge, or even diverge.
- One way to handle the issue of choosing step-sizes is to choose a different step-size at each step. If a given step-size leads to an increase in the value of the cost function, we have been too aggressive, so a better step-size would be half the current step-size. On the other hand, if the value of the function has decreased, the step-size could be larger. In each case we undo the current step and change the step-size.
- This heuristic guarantees monotonic convergence.

Gradient descent with momentum

- Gradient descent may be slow if the curvature of the function is such that the gradient descent steps hop between the walls of the valley of contours and approaches the optimum slowly.
- If we endow the optimization procedure with memory, we can improve convergence.
- We use an additional term in the step-update to remember what happened in the previous iteration, so that we can dampen oscillations and speed up convergence. This is a momentum term - the name momentum comes from a comparison to a rolling ball whose direction becomes more and more difficult to change as its velocity increases.
- In terms of equations we have
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^T + \alpha\Delta\boldsymbol{x}_i$.

Stochastic gradient descent

- The momentum term helps us deal with a nosiy estimate of the current gradient as the memory term remembers the past direction and prevents the current gradient estimate from leading the optimization procedure in a completely wrong direction.
- Why do we need to work with noisy or approximate gradients? This is because computing the true gradient may be computationally very demanding.
- We can deal with an approximate gradient and then use the momentum term to smooth out the noise.
- Stochastic gradient descent is a stochastic way of approximating the gradient of an objective function which is expressed as a sum of differentiable functions.

Stochastic gradient descent

- Let us consider a machine learning problem consisting of loss functions incurred at N data points. Let the loss function at the nth data point be $L_n(\theta)$, and let the total loss be $L(\theta) = \sum_{n=1}^{n=N} L_n(\theta)$. Here θ is the parameter vector of interest and the goal of machine learning is to find the parameter vector θ that minimizes the loss function.
- The standard gradient descent procedure is a batch optimization method in that the update step considers the gradient of the entire loss function $L(\theta)$., i,e $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma_i \nabla L(\boldsymbol{\theta}_i)^T = \boldsymbol{\theta}_i - \gamma_i \sum_{n=1}^{n=N} \nabla L_n(\boldsymbol{\theta}_i)^T$.
- This step is expensive since we may have many data points in the batch.

Stochastic gradient descent

- We can perform a mini-batch gradient descent by taking only a small subset of the data points.
- In the extreme case we can choose only one L_n to estimate the gradient.
- Why does taking only a subset of data points work? It works because we get an unbiased estimate of the true gradient.
- We can show that when the learning rate decreases at a suitable rate and some mild assumptions can be made, stochastic gradient descent almost surely converges to a local minimum.
- Estimating the gradient out of large mini-batches will lead to lower variance in the estimate, in contrast to estimating the gradient from small batch sizes.

Constrained optimization and Lagrange multipliers

- Consider the following problem: $\min_{\boldsymbol{x}} f(\boldsymbol{x}), f : \mathbb{R}^D \rightarrow \mathbb{R}$, subject to additional constraints - so we are looking at a minimization problem except that the set of all $\boldsymbol{x}$ over which minimization is performed is not all of $\mathbb{R}^D$.
- The constrained problem becomes $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0 \; \forall i 1, 2, \ldots m$.
- Since we have a method of finding a solution to the unconstrained optimization problem, one way to proceed now is to convert the given constrained optimization problem into an unconstrained one.
- We construct $J(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{i=m} \mathbf{1}(g_i(\boldsymbol{x}))$, where $\mathbf{1}(z) = 0$ for $z \leq 0$ and $\mathbf{1}(z) = \infty$ for $z > 0$.

Constrained optimization and Lagrange multipliers

- The formulation of $J(\boldsymbol{x})$ in the previous slide ensures that its value is infinity if any one of the constraints $g_i(\boldsymbol{x})$ is not satisfied. This ensures that the optimal solution to the unconstrained problem is the same as the constrained problem.
- The step-function is also difficult to optimize, and our solution is to replace the step-function by a linear function using Lagrange multipliers.
- We create the Lagrangian of the given constrained optimization problem as follows:
  $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{i=m} \lambda_i g_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$, where $\lambda_i \geq 0$ for all i.

Primal and dual problems

- The primal problem is $\min f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0, 1 \leq i \leq m$. Optimization is performed over the primal variables $\boldsymbol{x}$.
- The associated Lagrangian dual problem is $\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \mathfrak{D}(\boldsymbol{\lambda})$ subject to $\boldsymbol{\lambda} \geq 0$ where $\boldsymbol{\lambda}$ are dual variables.
- $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.
- The following minimax inequality holds over two arguments $\boldsymbol{x}, \boldsymbol{y}$: $\max_{\boldsymbol{y}} \min_{\boldsymbol{x}} \phi(\boldsymbol{x}, \boldsymbol{y}) \leq \min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \phi(\boldsymbol{x}, \boldsymbol{y})$.

Minimax inequality

- Why is this inequality true?
- Assume that $\boldsymbol{x}, \boldsymbol{y}$: $\max_{\boldsymbol{y}} \min_{\boldsymbol{x}} \phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}_A, \boldsymbol{y}_A)$ and $\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}_B, \boldsymbol{y}_B)$.
- Fixing $\boldsymbol{y}$ at $\boldsymbol{y}_A$ we see that the inner operation on the left hand side of the minimax inequality is a min operation over $\boldsymbol{x}$ and returns $\boldsymbol{x}_A$. Thus we have $\phi(\boldsymbol{x}_A, \boldsymbol{y}_A) \leq \phi(\boldsymbol{x}_B, \boldsymbol{y}_A)$.
- Fixing $\boldsymbol{x}$ at $\boldsymbol{x}_B$ we see that the inner operation on the right hand side of the minimax inequality is a max operation over $\boldsymbol{y}$ and returns $\boldsymbol{y}_B$. Thus we have $\phi(\boldsymbol{x}_B, \boldsymbol{y}_B) \geq \phi(\boldsymbol{x}_B, \boldsymbol{y}_A)$.
- From the above we conclude that $\phi(\boldsymbol{x}_B, \boldsymbol{y}_B) \geq \phi(\boldsymbol{x}_A, \boldsymbol{y}_A)$.

Minimax inequality

- The difference between $J(\boldsymbol{x})$ and the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is that the indicator function is relaxed to a linear function.
- When $\boldsymbol{\lambda} \geq 0$, the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a lower bound on $J(\boldsymbol{x})$.
- The maximum of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ is $J(\boldsymbol{x})$ - if the point $\boldsymbol{x}$ satisfies all the constraints $g_i(\boldsymbol{x}) \leq 0$, then the maximum of the Lagrangian is obtained at $\boldsymbol{\lambda} = 0$ and it is equal to $J(\boldsymbol{x})$. If one or more constraints is violated such that $g_i(\boldsymbol{x}) > 0$, then the associated Lagrangian coefficient λ_i can be taken to be ∞ so as to equal $J(\boldsymbol{x})$.

Minimax inequality

- From the previous slide, we have $J(\boldsymbol{x}) = \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.
- Our original constrained optimization problem boiled down to minimizing $J(\boldsymbol{x})$, in other words we are looking at $\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$
- Using the minimax inequality we see that $\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.
- This is known as weak duality. The inner part of the right hand side of the inequality is $\mathfrak{D}(\boldsymbol{\lambda})$, and the inequality above is the reason for setting up the associated Lagrangian dual problem for the original constrained optimization problem.

Lagrangian formulation

- In contrast to the original formulation $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is an unconstrained optimization problem for a given value of $\boldsymbol{\lambda}$.
- We observe that $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a point-wise minimum of affine functions and hence $\mathfrak{D}(\boldsymbol{\lambda})$ is concave even though $f()$ and $g()$ may be nonconvex.
- We have obtained a Lagrangian formulation for a constrained optimization problem where the constraints are inequalities. What happens when some constraints are equalities?

Modeling equality constraints

- Suppose the problem is $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0$ for all $1 \leq i \leq m$ and $h_j(\boldsymbol{x}) = 0$ for $1 \leq j \leq n$.
- We model the equality constraint $h_j(\boldsymbol{x}) = 0$ with two inequality constraints $h_j(\boldsymbol{x}) \geq 0$ and $h_j(\boldsymbol{x}) \leq 0$.
- The resulting Lagrange multipliers are then unconstrained.
- The Lagrange multipliers for the original inequality constraints are non-negative while those corresponding to the equality constraints are unconstrained.

Convex optimization

- We are interested in a class of optimization problems where we can guarantee global optimality.
- When $f()$, the objective function, is a convex function and $g()$ and $h()$ are convex functions, we have a convex optimization problem.
- In this setting we have strong duality - the optimal solution of the primal problem is equal to the optimal solution of the dual problem.
- What is a convex function?

Convex function

- First we need to know what is a convex set. A set C is a convex set if for any $x, y \in C$, $\theta x + (1-\theta)y \in C$.
- For any two points lying in the convex set, a line joining them lies entirely in the convex set.
- Let a function $f : \mathbb{R}^d \rightarrow R$ be a function whose domain is a convex set C.
- The function is a convex function if for any $\boldsymbol{x}, \boldsymbol{y} \in C$, $f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})$
- Another way of looking at a convex function is to use the gradient: for any two points $\boldsymbol{x}$ and $\boldsymbol{y}$, we have $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})$.

Example

- The negative entropy, a useful function in Machine Learning, is convex: $f(x) = xlog_2x$ for $x > 0$.
- First let us check if $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$. Take $x = 2$, $y = 4$, and $\theta = 0.5$ to get $f(0.5 * 2 + 0.5 * 4) = f(3) = 3log_23 \approx 4.75$. Then $\theta f(2) + (1-\theta)f(4) = 0.5 * 2log_22 + 0.5 * 4log_24 = log_232 = 5$. Therefore the convexity criterion is satisfied for these two points.
- Let us now use the gradient criterion. We have $\nabla x = log_2x + x\frac{1}{xlog_e2}$. Calculating $f(2) + \nabla f(2) * (4-2)$ gives $2log_22 + (log_22 + \frac{1}{log_e2} * 2 \approx 6.9$. We see that $f(4) = 4log_24 = 8$ which shows that the gradient criterion is also satisfied.

Linear programming

- Let us look at a convex optimization problem where the objective function and constraints are all linear.
- Such a convex optimization problem is called a linear programming problem.
- We can express a linear programming problem as $\min_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^{m \times 1}$.
- The Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is given by $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = c^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$ where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of non-negative Lagrangian multipliers.
- We can rewrite the Lagrangian as $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = (\boldsymbol{c} + \boldsymbol{A}^T \boldsymbol{\lambda})^T \boldsymbol{x} - \boldsymbol{\lambda}^T \boldsymbol{b}$.

Linear programming

- Taking the derivative of the Lagrangian with respect to $\boldsymbol{x}$ and setting it to zero we get $\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda} = 0$.
- Since $\mathfrak{D}(\lambda) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$, plugging in the above equation gives $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^T\boldsymbol{b}$.
- We would like to maximize $\mathfrak{D}(\boldsymbol{\lambda})$, subject to the constraint $\boldsymbol{\lambda} \geq 0$.
- Thus we end up with the following problem:

$$\begin{aligned} max_{\boldsymbol{\lambda}\in\mathbb{R}^m} \; & -\boldsymbol{\lambda}^T\boldsymbol{b} \\ \text{subject to } & \boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda} = 0 \\ & \boldsymbol{\lambda} \geq 0 \end{aligned}$$

Linear programming

- We can solve the original primal linear program or the dual one - the optimum in each case is the same.
- The primal linear program is in d variables but the dual is in m variables, where m is the number of constraints in the original primal program.
- We choose to solve the primal or dual based on which of m or d is smaller.

Quadratic programming

- We now consider the case of a quadratic objective function subject to affine constraints:

$$min_{\boldsymbol{x} \in \mathbb{R}^d} \frac{1}{2} \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x} \text{ subject to}$$

$$\boldsymbol{A} \boldsymbol{x} \leq \boldsymbol{b}$$

- Here $\boldsymbol{A} \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m, c \in \mathbb{R}^d$

Quadratic programming

- The Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is given by $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$.
- Rearranging the above we have $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{x} - \boldsymbol{\lambda}^T\boldsymbol{b}$
- Taking the derivative of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ and setting it equal to zero gives $\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) = 0$.
- If we take $\boldsymbol{Q}$ to be invertible, we have $\boldsymbol{x} = \boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})$.
- Plugging this value of $\boldsymbol{x}$ into $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ gives us $\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$.
- This gives us the dual optimization problem: $\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$ subject to $\boldsymbol{\lambda} \geq \boldsymbol{0}$.