PART I. OPTIMIZATION: CLASSICAL APPROACHES
(LECTURE 10)

Shpilev Petr Valerievich
Faculty of Mathematics and Mechanics, SPbU

September, 2025

Санкт-Петербургский государственный университет

Comments

In this lecture, we explore advanced algorithms for solving convex and nonconvex quadratic programming (QP) problems. We begin with the projected conjugate gradient (CG) method, discussing its projection computation and the use of general preconditioners to enhance convergence. We then shift focus to optimality conditions for convex QPs and address challenges in nonconvex settings, including degeneracy. The lecture introduces active-set methods for convex QPs, with a detailed examination of primal active-set algorithms, including step computation, iteration, and termination criteria. We explore the concept of feasible and descent directions, as well as optimality checks and convergence analysis. Through practical examples, we demonstrate the application of primal active-set methods, discussing how to initialize feasible points and apply the Big M method. Finally, we cover the QR update for adding constraints, its relationship to the Hessian, and key properties of the active-set method to ensure reliable convergence.

Projected CG Method

Algorithm: Projected CG

1: Choose initial x with $Ax = b$;
2: Compute $r = Gx + c$, $g = Pr$, $d = -g$;
3: for iterations until $r^T Pr$ is is smaller than a prescribed tolerance do
4: $\quad \alpha \leftarrow r^T g / d^T G d$;
5: $\quad x \leftarrow x + \alpha d$;
6: $\quad r^+ \leftarrow r + \alpha G d$;
7: $\quad g^+ \leftarrow Pr^+$;
8: $\quad \beta \leftarrow (r^+)^T g^+ / r^T g$;
9: $\quad d \leftarrow -g^+ + \beta d$;
10: $\quad g \leftarrow g^+$, $r \leftarrow r^+$;

- *Preconditioned residual* $g^+ = Pr^+$ lies in null space of A, ensuring $Ad = 0$ and $Ax = b$.
- Requires $Z^T G Z$ and $Z^T H Z$ positive definite for well-defined iterations.
- Preconditioner H: e.g., $\mathrm{diag}(|G_{ii}|)$, I, or block diagonal submatrix of G.

Projected CG iterates in null space for efficient QP solution.

Comments

The projected conjugate gradient algorithm is essentially a direct translation of the standard preconditioned conjugate gradient method into the setting where feasibility with respect to the constraints must always be maintained. The structure of the algorithm reflects this requirement. We begin with an initial point x that already satisfies the linear constraints $Ax = b$. This is crucial, because all subsequent iterates must remain feasible.

At each step, the residual r is computed as $Gx + c$, which represents the violation of the optimality conditions. However, rather than working with r directly, we project it into the null space using $g = Pr$. This projected residual serves as the effective search gradient, ensuring that all directions are feasible. The initial search direction is then taken as $-g$.

The iteration follows the usual conjugate gradient structure: a step size α is chosen based on the ratio of $r^T g$ to $d^T G d$, the solution is updated by $x + \alpha d$, and the residual is correspondingly updated. The crucial difference lies in the computation of the new preconditioned residual g^+, which is again obtained by projecting the new residual r^+ into the null space. The coefficient β is then computed in the familiar way, and the new search direction is formed so as to preserve conjugacy.

The important theoretical property here is that g^+ always lies in the null space of A. This guarantees that both the search direction d and the iterates x remain feasible throughout the process. For well-defined behavior, we require that both $Z^T G Z$ and $Z^T H Z$ are positive definite, ensuring that all denominators and projections are meaningful.

In practice, the choice of preconditioner H has a strong influence on performance. Simple options such as the identity matrix, a diagonal scaling with absolute values of G's diagonal entries, or block diagonal approximations of G can already yield significant acceleration. Thus, the projected conjugate gradient method provides an efficient and robust way to solve quadratic programs with equality constraints by embedding feasibility directly into the iterative process.

Projected CG: Computing the Projection

Projected CG uses $P = Z(Z^T H Z)^{-1} Z^T$. For $H = I$, orthogonal projection is:

$$P_I = Z(Z^T Z)^{-1} Z^T = I - A^T (AA^T)^{-1} A.$$

Compute $g^+ = P_I r^+$ in two ways:

- Normal equations: Solve $AA^T v^+ = Ar^+$, then $g^+ = r^+ - A^T v^+$, using Cholesky factorization of AA^T.
- Augmented system: Solve

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix}$$

  via symmetric indefinite factorization.

- No explicit Z needed; only matrix-vector products with P_I.
- Normal equations: efficient for well-conditioned AA^T.
- Augmented system: robust for large, sparse systems.

Projection computed efficiently without null-space basis Z.

Comments

The computation of the projection in the projected conjugate gradient method can be achieved without explicitly constructing the null-space basis. In the simplest case, when the preconditioning matrix H is equal to the identity, the projection operator reduces to an orthogonal projection. This projection can be written in two equivalent forms. The first uses the null-space matrix, but a more practical expression avoids it entirely, namely $I - A^T(AA^T)^{-1}A$. This formula shows that the projection can be realized using only operations involving A and its transpose.

There are two standard procedures to apply the projection to a residual vector. The first relies on solving the so-called normal equations. One computes Ar^+, solves the system $AA^T v = Ar^+$, and then recovers the projection as $r^+ - A^T v$. Since the matrix AA^T is symmetric and positive definite, a Cholesky factorization can be precomputed once and reused in each iteration, making this route efficient when the conditioning is reasonable.

The second approach is to work with the augmented system. In this case, one solves a larger symmetric indefinite system involving identity, A, and its transpose. This approach is more robust, especially when dealing with large sparse problems where direct use of the normal equations may lead to numerical instability.

The key message is that the projection can be computed entirely implicitly. There is no need to form the basis matrix Z, nor to handle it explicitly. Only matrix–vector products with A and the corresponding factorizations are required, which makes the method suitable for practical large-scale constrained optimization.

Projected CG: General Preconditioner

Compute $g^+ = Pr^+$ with $P = H^{-1}(I - A^T(AH^{-1}A^T)^{-1}AH^{-1})$ when H is nonsingular, or solve (when $z^THz \neq 0$ for all nonzero z with $Az = 0$):

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix}.$$

- Constraint preconditioner ($H \neq G$) requires Z^THZ positive definite.
- Ideal: $H = G$.
- Initial point x with $Ax = b$ via:

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \text{ or } \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

- No explicit Z; uses factorizations of AA^T or system matrices.
- Backsolve computes x efficiently from projection factorizations.
- Iterative refinement mitigates round-off errors in g^+.

General preconditioner enables robust, implicit projection.

Comments

The use of a general preconditioner greatly expands the flexibility of the projected conjugate gradient framework. In this setting, the projection operator is defined with the preconditioner matrix H, which is assumed to be nonsingular. The expression for the projection then becomes $H^{-1}(I - A^T(AH^{-1}A^T)^{-1}AH^{-1})$. Although this formula appears complicated, in practice the projection is again never formed explicitly. Instead, one applies it by solving linear systems.

A robust way to implement the projection is to solve the augmented saddle-point system consisting of the block matrix with H and A^T in the first row and A with zero in the second row. The unknowns are the projected vector and an auxiliary multiplier. This formulation is attractive because it naturally enforces the constraints and incorporates the preconditioner in a symmetric and stable way.

It is important that the reduced matrix Z^THZ be positive definite, which ensures that the method works correctly. The ideal choice is to set $H = G$, the Hessian of the quadratic objective. In that case, the projection aligns perfectly with the natural metric of the problem, and convergence is typically accelerated.

In addition, the same factorization used for projection can be exploited to compute a feasible initial point. One can solve either the identity-based augmented system or the H-based version with the right-hand side containing the constraint vector b. This approach allows one to find a starting x that already satisfies the equality constraints.

Finally, iterative refinement can be applied to improve numerical stability, correcting small errors that accumulate in the projection step and ensuring accurate progress of the algorithm.

For QPs with equality and inequality constraints:

- Active-set (1970s): Suits small/medium problems, detects unboundedness/infeasibility, estimates optimal active set.
- Interior-point (1990s): Ideal for large problems, less suited for related QPs.
- Gradient projection: Efficient for bound-constrained QPs.

Definition: Active Set

The active set at a solution x^* is the following set: $\mathcal{A}(x^*) = \{i \in \mathcal{E} \cup \mathcal{I} \mid a_i^T x^* = b_i\}$.

Lagrangian: $\mathcal{L}(x, \lambda) = \frac{1}{2} x^T G x + x^T c - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i (a_i^T x - b_i)$.

KKT conditions at x^* with λ_i^*:

$$
\begin{aligned}
& Gx^* + c - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i^* a_i = 0, \\
& a_i^T x^* = b_i, \ \text{for } i \in \mathcal{A}(x^*), \\
& a_i^T x^* > b_i, \ \text{for } i \in \mathcal{I} \setminus \mathcal{A}(x^*), \\
& \lambda_i^* \geq 0, \quad \text{for } i \in \mathcal{I} \cap \mathcal{A}(x^*), \\
& \lambda_i^* (a_i^T x^* - b_i) = 0, \ \text{for } i \in \mathcal{I} \cup \mathcal{E}.
\end{aligned}
$$

Comments

Quadratic programming with both equality and inequality constraints has motivated the development of several algorithmic families, each with strengths in particular regimes. One of the earliest approaches is the active-set method, originating in the 1970s. This strategy is well suited for small to medium sized problems. Its main advantage lies in its ability to detect infeasibility and unboundedness while systematically building an estimate of the active set of constraints at the solution. By maintaining and updating this active set, the method effectively reduces the problem to a sequence of equality-constrained quadratic programs.

In contrast, interior-point methods became dominant in the 1990s. They are particularly effective for very large and sparse quadratic programs. Their power comes from a polynomial complexity bound and robustness in treating large-scale linear constraints. However, their drawback is that they are less convenient when many related quadratic programs must be solved, since each new instance may require a fresh factorization.

Another important class is gradient projection methods, which are especially efficient for bound-constrained problems. In these cases, the constraints simply restrict variables to lie within intervals, and projection onto the feasible set can be done by simple clipping operations. This makes such methods lightweight and scalable.

The concept of the active set plays a central role. At an optimal solution, the active set is defined as the set of all equality constraints together with those inequality constraints that are satisfied at equality. With this definition, the Karush–Kuhn–Tucker conditions can be stated compactly. The Lagrangian is formed from the quadratic objective and a weighted combination of all constraints. The KKT conditions then characterize stationarity, primal feasibility, and dual feasibility, and for inequality constraints they include the nonnegativity of the associated multipliers and complementarity.

KKT conditions for QP hold without LICQ, e.g., under linear constraints, satisfied in quadratic programming. For convex QP (G positive semidefinite), KKT conditions are sufficient for global optimality.

Theorem 37 (Sufficiency for Convex QP)

If x^* satisfies KKT conditions with λ_i^*, $i \in \mathcal{A}(x^*)$, and G is positive semidefinite, then x^* is a global solution of the QP.

Proof: For any feasible x, $a_i^T(x - x^*) = 0$ for $i \in \mathcal{E}$, $a_i^T(x - x^*) \geq 0$ for $i \in \mathcal{A}(x^*) \cap \mathcal{I}$. Thus:

$$(x - x^*)^T(Gx^* + c) = \sum_{i \in \mathcal{E}} \lambda_i^* a_i^T(x - x^*) + \sum_{i \in \mathcal{A}(x^*) \cap \mathcal{I}} \lambda_i^* a_i^T(x - x^*) \geq 0.$$

Hence:

$$q(x) = q(x^* + (x - x^*)) = q(x^*) + (x - x^*)^T(Gx^* + c) + \frac{1}{2}(x - x^*)^T G(x - x^*) \geq$$
$$q(x^*) + \frac{1}{2}(x - x^*)^T G(x - x^*) \geq q(x^*),$$

since G is positive semidefinite, proving x^* is a global solution. □

Note: KKT conditions ensure global optimality for convex QPs.

Comments

The Karush–Kuhn–Tucker conditions serve as the fundamental optimality criterion for quadratic programming. Remarkably, in the quadratic case these conditions hold under very mild assumptions, even without requiring the linear independence constraint qualification. For convex quadratic programs, where the Hessian matrix G is positive semidefinite, the KKT conditions are not only necessary but also sufficient for global optimality.

Let's prove this theorem. Suppose a candidate point x^* and multipliers λ^* satisfy the KKT system. Consider any other feasible point x. For each equality constraint, the inner product of its coefficient vector with the difference $x - x^*$ vanishes. For each inequality that is active at x^*, the corresponding multiplier is nonnegative, and the inner product with $x - x^*$ is also nonnegative. Combining these relations shows that the product of $x - x^*$ with the gradient of the objective at x^* is nonnegative.

Next, expand the quadratic objective at $x^* + (x - x^*)$. By direct substitution we immediately obtain that the value $q(x)$ equals $q(x^*)$ plus the inner product of $x - x^*$ with the gradient at x^* plus $\frac{1}{2}(x - x^*)^T G(x - x^*)$. Because the Hessian is positive semidefinite, the last term is always nonnegative. Since the linear term was also shown to be nonnegative, the entire expression is greater than or equal to $q(x^*)$. This inequality demonstrates that x^* achieves the minimum among all feasible points, and therefore is a global solution. The theorem is proved.

Thus, in convex quadratic programming the KKT conditions not only describe optimality but actually guarantee it, providing a clean and powerful characterization of solutions.

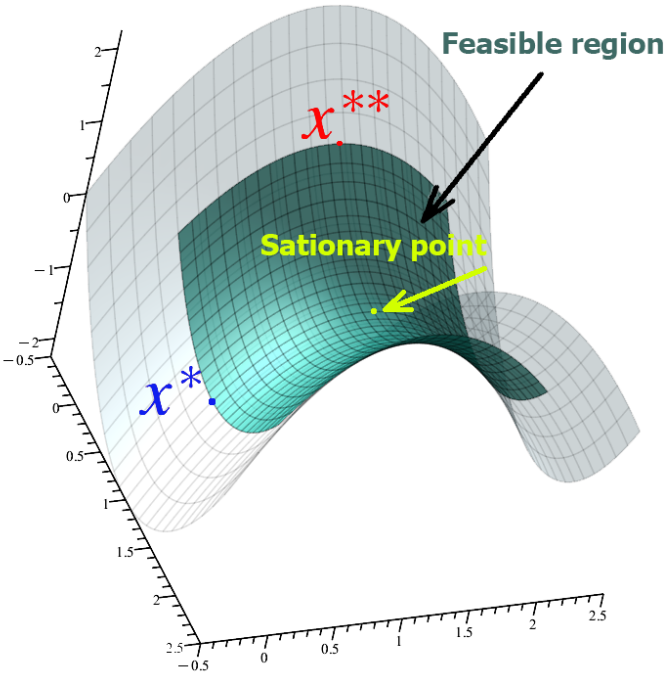


Figure: Here we have plotted the feasible region and the graph of a quadratic objective q(x) in which G has one positive and one negative eigenvalue. Note that x^{**} is a local maximizer, x^* a local minimizer, and the center of the region is a stationary point.

Comments

Quadratic programming becomes much more intricate once convexity is lost. In a convex problem, the Hessian matrix G is positive semidefinite, guaranteeing that every local minimizer is also a global minimizer. But in the nonconvex case, G has at least one negative eigenvalue, and the landscape of the objective function becomes significantly more complicated. A single stationary point is no longer sufficient to characterize the solution, because stationary points may correspond to local minima, local maxima, or even saddle points.

To illustrate, imagine a quadratic function of two variables where the Hessian has one positive and one negative eigenvalue. In this case the graph of the function forms a saddle-shaped surface. At certain feasible points, such as x^*, the point is indeed a local minimizer, while at another feasible point, say x^{**}, the point turns out to be a local maximizer. The very center of the feasible region may even be a stationary point without being an extremum.

This lack of convexity raises several challenges. First, optimality conditions such as the Karush–Kuhn–Tucker system no longer guarantee global solutions. They can only identify stationary points, and further analysis is required to classify their nature. Second, algorithmic strategies like active-set or interior-point methods may converge to local minima that are not globally optimal. Finally, the geometry of feasible regions interacts with the curvature of the quadratic objective in subtle ways, often making global optimization extremely difficult.

For these reasons, nonconvex quadratic programming is generally considered computationally hard. Specialized algorithms exist, for example branch-and-bound or cutting-plane techniques, but they are far less efficient than methods available in the convex case. This contrast underlines why convexity is such a central property in optimization theory.

Degeneracy in Quadratic Programming

Degeneracy occurs when:

- Active constraint gradients a_i, $i \in \mathcal{A}(x^*)$, are linearly dependent.
- Strict complementarity fails: $\exists i \in \mathcal{A}(x^*) \cap \mathcal{I}$ with $\lambda_i^* = 0$ (weakly active constraint).

Example: $\min x_1^2 + (x_2 + 1)^2$ s.t. $x \geq 0$, solution $x^* = 0$, both constraints active, $\lambda_1^* = 0$, violating strict complementarity.

- Linear dependence causes rank-deficient matrices, complicating step computation.
- Weakly active constraints lead to zigzagging in active-set/gradient projection methods.
- Safeguards prevent algorithmic oscillation near weakly active constraints.

Degeneracy challenges numerical stability and constraint identification.

Comments

Another important phenomenon in quadratic programming is degeneracy. Degeneracy occurs when the structure of the active set at a solution point prevents algorithms from behaving in a straightforward way. There are two primary causes of degeneracy.

The first arises when the gradients of the active constraints are linearly dependent. At the solution, the active set is defined by equality constraints and those inequalities that are tight. If the associated gradient vectors are linearly dependent, then the matrix used in computing search directions becomes rank-deficient, and algorithms may struggle to define unique steps.

The second source of degeneracy is the failure of strict complementarity. Strict complementarity means that every active inequality constraint at the solution has a strictly positive multiplier. If instead some active inequality has multiplier equal to zero, the constraint is only weakly active. This situation leads to ambiguous behavior: algorithms such as active-set or gradient projection methods may oscillate near weakly active constraints, producing a zigzagging path instead of steady convergence.

A concrete example makes this clear. Consider minimizing $x_1^2 + (x_2 + 1)^2$ subject to nonnegativity of both variables. The solution occurs at the origin, where both constraints are active. However, the multiplier for the first constraint is zero, violating strict complementarity.

Degeneracy has practical consequences. It complicates the identification of the correct active set, and it challenges numerical stability, especially when iterative updates require solving ill-conditioned systems. Modern implementations therefore incorporate safeguards to detect and control oscillations. Although degeneracy cannot be avoided in principle, careful algorithmic design ensures that optimization methods remain robust in its presence.

For convex QPs (G positive semidefinite), solve:

$$\min_x q(x) = \frac{1}{2}x^T Gx + x^T c \quad \text{s.t.} \quad a_i^T x = b_i,\ i \in \mathcal{E}, \quad a_i^T x \geq b_i,\ i \in \mathcal{I}.$$

Active-set methods iteratively estimate optimal active set $\mathcal{A}(x^*)$. Primal methods:

- Maintain feasible iterates, decrease $q(x)$.
- Solve subproblems with working set $\mathcal{W}_k \subseteq \mathcal{E} \cup \mathcal{I}$ as equalities.
- Ensure a_i, $i \in \mathcal{W}_k$, are linearly independent.

- Analogous to simplex method: Adjust $\mathcal{W}_k$ using gradients and multipliers.
- Challenge: Identify $\mathcal{A}(x^*)$ without prior knowledge.
- Iterates may not be vertices, unlike linear programming.

Primal active-set methods iteratively refine $\mathcal{W}_k$ for QP solutions.

Comments

In convex quadratic programming, where the Hessian matrix G is positive semidefinite, active-set methods form one of the most classical algorithmic approaches. The idea is to iteratively build an approximation of the optimal active set, denoted by $\mathcal{A}(x^*)$. At each iteration, the algorithm selects a working set, denoted $\mathcal{W}_k$, which is a subset of the equality and inequality constraints. These constraints are treated as equalities, and the resulting subproblem is solved.

The working set must be chosen carefully. The gradients of its constraints, denoted by vectors a_i, must be linearly independent to avoid rank deficiencies. Within this reduced system, the algorithm computes a feasible step that decreases the quadratic objective function. In this sense, the procedure resembles the simplex method for linear programming, where the basis of constraints is modified step by step to move closer to the optimum.

However, there are important differences compared to linear programming. In quadratic programming, iterates generated by active-set methods are not necessarily vertices of the feasible polyhedron. Instead, they may lie along edges or faces, reflecting the curved nature of the quadratic objective. The main challenge is that the optimal active set is unknown in advance, so the algorithm must discover it dynamically, adjusting the working set based on gradients and multipliers.

Despite these complications, active-set methods are particularly effective for small to medium-sized convex quadratic programs. They maintain feasibility of all iterates, provide systematic progress toward the solution, and offer a clear geometric interpretation. For these reasons, active-set strategies remain widely studied and applied, especially when reliability and interpretability are valued.

Key Idea Primal active-set methods iteratively refine working set $\mathcal{W}_k$ to approximate optimal active set $\mathcal{A}(x^*)$.

Use $\mathcal{W}_k \subseteq \mathcal{E} \cup \mathcal{I}$ at iterate x_k, with linearly independent gradients a_i, $i \in \mathcal{W}_k$. If x_k is not optimal in $\mathcal{W}_k$, solve subproblem:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (G x_k + c)^T p \\ \text{s.t.} \quad & a_i^T p = 0, \quad i \in \mathcal{W}_k, \end{aligned}$$

where $p = x - x_k$, derived from $q(x_k + p) = \frac{1}{2} p^T G p + (G x_k + c)^T p + \rho_k$, with $\rho_k = \frac{1}{2} x_k^T G x_k + c^T x_k$ which is independent of p.

- Subproblem imposes $\mathcal{W}_k$ as equalities, ignores other constraints.
- Linear independence of a_i ensures solvable subproblem.
- Step p reduces $q(x)$ while maintaining feasibility.

Working set $\mathcal{W}_k$ guides feasible steps in QP optimization.

Comments

The primal active-set method refines the working set in a systematic way, making it a powerful tool for solving convex quadratic programs. At a given iterate, denoted x_k, the algorithm maintains a working set $\mathcal{W}_k$ consisting of linearly independent constraints. If x_k is not yet optimal with respect to this working set, a subproblem is solved to find a feasible step.

This subproblem is defined in terms of the search direction p, where $p = x - x_k$. The objective of the subproblem is $\frac{1}{2} p^T G p + (G x_k + c)^T p$. The constraints of the subproblem enforce that $a_i^T p = 0$ for every constraint in the working set. This construction ensures that the step p respects the currently assumed active constraints.

By definition, the quadratic objective function at $x_k + p$ equals $\frac{1}{2} p^T G p + (G x_k + c)^T p + \rho_k$, where ρ_k represents the part of the objective that depends only on x_k. Since ρ_k is independent of p, the subproblem can be solved by focusing only on the first two terms.

The step p computed in this manner reduces the value of the objective function while maintaining feasibility with respect to the active set. If the step remains feasible for all constraints, it is accepted; otherwise, the working set is updated by adding or removing constraints. Through repeated iterations, the algorithm converges toward the true optimal active set and thereby finds the solution of the quadratic program.

Primal Active-Set Methods: Step Computation

Solve QP subproblem for step p_k:

$$\min_p \quad \frac{1}{2} p^T G p + (G x_k + c)^T p$$
$$\text{s.t.} \quad a_i^T p = 0, \quad i \in \mathcal{W}_k.$$

For $i \in \mathcal{W}_k$, constraints remain satisfied: $a_i^T(x_k + \alpha p_k) = a_i^T x_k = b_i$.
Update:

$$x_{k+1} = x_k + \alpha_k p_k, \quad \alpha_k \in [0, 1],$$

where $\alpha_k \stackrel{\text{def}}{=} \min\left(1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k}\right)$.

- $p_k \neq 0$ solved using equality-constrained QP techniques (G positive definite).
- α_k ensures feasibility for $i \notin \mathcal{W}_k$ when $a_i^T p_k < 0$ (since the inequality holds: $a_i^T(x_k + \alpha_k p_k) \geq b_i$).
- Constraints with $a_i^T p_k \geq 0$ for some $i \notin \mathcal{W}_k$ remain satisfied for $\alpha_k \geq 0$.

Step p_k and α_k maintain feasibility and reduce $q(x)$.

Comments

In primal active-set methods, each iteration begins by solving a quadratic programming subproblem to compute a search direction, denoted p_k. The subproblem minimizes $\frac{1}{2} p^T G p + (G x_k + c)^T p$, subject to the constraints in the current working set $\mathcal{W}_k$ treated as equalities. This guarantees that all constraints currently assumed active remain satisfied along the step. Specifically, for each $i \in \mathcal{W}_k$, $a_i^T(x_k + \alpha_k p_k) = b_i$.

The computed step p_k is then scaled by a step length α_k. If taking the full step preserves feasibility for all constraints, α_k is set to 1. Otherwise, α_k is determined by the most restrictive constraint outside the working set. In particular, for each $i \notin \mathcal{W}_k$ where $a_i^T p_k$ is negative, α_k is taken as the minimum of the ratio of $b_i - a_i^T x_k$ over $a_i^T p_k$. This ensures that no constraint is violated along the step.

This mechanism guarantees that every iterate remains feasible while ensuring progress in reducing the objective function $q(x)$. If a constraint is already satisfied and its $a_i^T p_k$ is nonnegative, it remains satisfied for any nonnegative α_k. By combining p_k and α_k in this way, the algorithm moves along a feasible path in the search space, systematically decreasing the quadratic objective while maintaining all required equalities and inequalities.

Constraints $i \notin \mathcal{W}_k$ for which step-length α_k is minimized are called *blocking constraints*.

If $\alpha_k = 1$ and no new constraints active at $x_k + \alpha_k p_k$, there are no blocking constraints.
If $\alpha_k < 1$, add one of the blocking constraints to $\mathcal{W}_{k+1}$.
Note: $\alpha_k = 0$ possible if a constraint i is active at x_k, $i \notin \mathcal{W}_k$.

Iterate by adding constraints to $\mathcal{W}_k$ until a point $\hat{x}$ is reached where $p = 0$ is the solution to the QP subproblem.

- This means $\hat{x}$ minimizes $q(x)$ over $\hat{\mathcal{W}}$ and the first KKT optimality condition for this subproblem is satisfied:

$$G\hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i \text{ (for some Lagrange multipliers } \hat{\lambda}_i)$$

  since we set Lagrange multipliers for constraints not in $\hat{\mathcal{W}}$ to zero.
- The step length α_k ensures feasibility, so $\hat{x}$ also satisfies the second and the third KKT conditions as well.

Comments

Once a candidate solution $\hat{x}$ is obtained, its optimality is verified by examining the signs of the Lagrange multipliers associated with inequality constraints in the working set, denoted $\hat{\lambda}_i$ for $i \in \hat{\mathcal{W}} \cap \mathcal{I}$. If all of these multipliers are nonnegative, the fourth KKT condition is satisfied, indicating that $\hat{x}$ is a KKT point. Because G is positive semidefinite, $\hat{x}$ is then a global solution. If G is strictly positive definite, the global solution is unique.

If, however, any $\hat{\lambda}_j$ is negative for $j \in \hat{\mathcal{W}} \cap \mathcal{I}$, the fourth KKT condition fails. In this case, the algorithm removes one of these constraints from the working set and resolves the QP subproblem for a new search direction p. This new step is feasible with respect to the dropped constraint, ensuring progress while maintaining feasibility for all remaining constraints.

Throughout the iterations, linear independence of the gradients a_i for $i \in \mathcal{W}_k$ is maintained. This is critical because it guarantees that the equality-constrained subproblem remains solvable. By continually adjusting $\mathcal{W}_k$ based on the signs of Lagrange multipliers, the algorithm navigates toward a globally optimal solution. The process balances between expanding the active set when a blocking constraint appears and contracting it when a negative multiplier indicates the corresponding constraint is unnecessarily restrictive.

In this way, the primal active-set method converges to a feasible point where all KKT conditions are satisfied. It provides a clear and constructive approach to convex quadratic programming, combining step computation, feasibility enforcement, and multiplier-based checks into a coherent algorithmic framework.

Primal Active-Set Methods: Checking Optimality

Check signs of Lagrange multipliers $\hat{\lambda}_i$ for inequality constraints $i \in \hat{\mathcal{W}} \cap \mathcal{I}$ at $\hat{x}$.

- If all $\hat{\lambda}_i \geq 0$, the fourth KKT condition is satisfied, so $\hat{x}$ is a KKT point.
- Since G is positive semidefinite, $\hat{x}$ is a global solution. If G is positive definite, $\hat{x}$ is the unique global solution (by Theorem 37).

If any $\hat{\lambda}_j < 0$ for $j \in \hat{\mathcal{W}} \cap \mathcal{I}$, the fourth KKT condition fails.

- Remove one such j from $\mathcal{W}_{k+1}$ and solve a new QP subproblem for the next step p.
- The new step p is feasible for the dropped constraint.

Constraint gradients a_i, $i \in \mathcal{W}_k$, are linearly independent (maintained throughout iterations).

Adjust $\mathcal{W}_k$ based on $\hat{\lambda}_i$ signs to find global solution $\hat{x}$.

Comments

Once a candidate solution $\hat{x}$ is obtained, its optimality is verified by examining the signs of the Lagrange multipliers associated with inequality constraints in the working set, denoted $\hat{\lambda}_i$ for $i \in \hat{\mathcal{W}} \cap \mathcal{I}$. If all of these multipliers are nonnegative, the fourth KKT condition is satisfied, indicating that $\hat{x}$ is a KKT point. Because G is positive semidefinite, $\hat{x}$ is then a global solution. If G is strictly positive definite, the global solution is unique.

If, however, any $\hat{\lambda}_j < 0$ for $j \in \hat{\mathcal{W}} \cap \mathcal{I}$, the fourth KKT condition fails. In this case, the algorithm removes one of these constraints from the working set and resolves the QP subproblem for a new search direction p. This new step is feasible with respect to the dropped constraint, ensuring progress while maintaining feasibility for all remaining constraints.

Throughout the iterations, linear independence of the gradients a_i for $i \in \mathcal{W}_k$ is maintained. This is critical because it guarantees that the equality-constrained subproblem remains solvable. By continually adjusting $\mathcal{W}_k$ based on the signs of Lagrange multipliers, the algorithm navigates toward a globally optimal solution. The process balances between expanding the active set when a blocking constraint appears and contracting it when a negative multiplier indicates the corresponding constraint is unnecessarily restrictive.

In this way, the primal active-set method converges to a feasible point where all KKT conditions are satisfied. It provides a clear and constructive approach to convex quadratic programming, combining step computation, feasibility enforcement, and multiplier-based checks into a coherent algorithmic framework.

Theorem 38: Feasible direction

Suppose that the point $\hat{x}$ satisfies first-order conditions for the equality-constrained subproblem with working set $\hat{\mathcal{W}}$; that is,

$$G\hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i$$

is satisfied along with $a_i^T \hat{x} = b_i$ for all $i \in \hat{\mathcal{W}}$. Suppose, too, that the constraint gradients a_i, $i \in \hat{\mathcal{W}}$, are linearly independent and that there is an index $j \in \hat{\mathcal{W}}$ such that $\hat{\lambda}_j < 0$. Let p be the solution obtained by dropping the constraint j and solving the following subproblem:

$$\begin{aligned} \min_p \quad & \frac{1}{2} p^T G p + (G\hat{x} + c)^T p, \\ \text{subject to} \quad & a_i^T p = 0, \quad \text{for all } i \in \hat{\mathcal{W}} \text{ with } i \neq j. \end{aligned}$$

Then p is a feasible direction for constraint j, that is, $a_j^T p \geq 0$. Moreover, if p satisfies second-order sufficient conditions for the subproblem, then we have that $a_j^T p > 0$, and that p is a descent direction for $q(\cdot)$.

Comments

The concept of feasible directions plays an important role in refining the working set. Suppose a candidate point $\hat{x}$ satisfies the first-order conditions for the equality-constrained subproblem defined by the working set $\hat{\mathcal{W}}$, and that the constraint gradients a_i for $i \in \hat{\mathcal{W}}$ are linearly independent. If a multiplier $\hat{\lambda}_j$ associated with some constraint $j \in \hat{\mathcal{W}}$ is negative, the algorithm can generate a feasible direction by temporarily dropping constraint j and solving a modified subproblem.

The subproblem minimizes $\frac{1}{2} p^T G p + (G\hat{x} + c)^T p$, subject to $a_i^T p = 0$ for all $i \in \hat{\mathcal{W}}$ excluding j. The resulting vector p satisfies $a_j^T p \geq 0$, making it a feasible direction for the previously active constraint. If second-order sufficient conditions hold, then $a_j^T p > 0$, meaning that moving along p increases satisfaction of the previously restrictive constraint and reduces the quadratic objective.

This property guarantees that the working set can be adjusted systematically. Negative multipliers indicate which constraints are too restrictive, while feasible directions allow the algorithm to make progress without violating any other active constraints. The step along p is both feasible and a descent direction, ensuring that the objective function decreases. This mechanism forms the foundation for the removal of constraints with negative multipliers and helps drive the iteration toward the global optimum.

Proof:

Since p solves the subproblem:

$$\begin{aligned} \min_{p} \quad & \frac{1}{2} p^T G p + (G\hat{x} + c)^T p, \\ \text{s.t.} \quad & a_i^T p = 0, \quad \forall i \in \hat{\mathcal{W}}, i \neq j, \end{aligned}$$

there exist multipliers $\tilde{\lambda}_i$, for all $i \in \hat{\mathcal{W}}$, $i \neq j$, such that:

$$\sum_{i \in \hat{\mathcal{W}}, i \neq j} \tilde{\lambda}_i a_i = Gp + (G\hat{x} + c).$$

- By second-order necessary conditions, if Z is a null-space basis for the matrix $\left[a_i^T\right]_{i \in \hat{\mathcal{W}}, i \neq j}$, then $Z^T G Z$ is positive semidefinite.
- Since $p = Zp_z$ for some p_z, it follows that $p^T G p \geq 0$.
- Subtracting $G\hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i$ from the above, we get:

$$\sum_{i \in \hat{\mathcal{W}}, i \neq j} (\tilde{\lambda}_i - \hat{\lambda}_i) a_i - \hat{\lambda}_j a_j = Gp.$$

Comments

To justify that dropping a constraint with a negative multiplier produces a feasible direction, consider the subproblem solved for p. This subproblem minimizes $\frac{1}{2} p^T G p + (G\hat{x} + c)^T p$, subject to all constraints in $\hat{\mathcal{W}}$ except the dropped index j treated as equalities. By optimality of this subproblem, there exist Lagrange multipliers, denoted $\tilde{\lambda}_i$ for $i \in \hat{\mathcal{W}}$ excluding j, such that the sum over i of $\tilde{\lambda}_i a_i$ equals $Gp + G\hat{x} + c$.

Applying second-order necessary conditions, let Z be a basis for the null space of the matrix formed by a_i^T for $i \in \hat{\mathcal{W}}$ excluding j. Then $Z^T G Z$ is positive semidefinite. Since p lies in the column space of Z, it follows that $p^T G p \geq 0$. Subtracting the first-order condition at $\hat{x}$, namely $G\hat{x} + c = \sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i a_i$, we obtain the equation: the sum over $i \in \hat{\mathcal{W}}$ excluding j of $(\tilde{\lambda}_i - \hat{\lambda}_i) a_i - \hat{\lambda}_j a_j = Gp$. This establishes a foundation for showing that $a_j^T p \geq 0$, proving that p is a feasible direction for constraint j.

Primal Active-Set Methods: Proof of Feasible Direction (Cont.)

By taking inner products of both sides with p and using $a_i^T p = 0$ for all $i \in \hat{\mathcal{W}}$, $i \neq j$, we have:

$$-\hat{\lambda}_j a_j^T p = p^T G p.$$

Since $p^T G p \geq 0$ and $\hat{\lambda}_j < 0$, it follows that $a_j^T p \geq 0$.

- If second-order sufficient conditions hold, $Z^T G Z$ is positive definite. Then $a_j^T p = 0$ only if $p^T G p = p_z^T Z^T G Z p_z = 0$, so $p_z = 0$ and $p = 0$.
- If $p = 0$, then $\sum_{i \in \hat{\mathcal{W}}, i \neq j} (\tilde{\lambda}_i - \hat{\lambda}_i) a_i - \hat{\lambda}_j a_j = 0$. By linear independence of a_i, $i \in \hat{\mathcal{W}}$, we get $\hat{\lambda}_j = 0$, a contradiction.
- Thus, $p^T G p > 0$, so $a_j^T p > 0$ when second-order sufficient conditions hold.

The claim that p is a descent direction for $q(\cdot)$ will be proved further. □

While any j with $\hat{\lambda}_j < 0$ yields a direction p for progress, we often choose the most negative $\hat{\lambda}_j$.

- Sensitivity analysis shows the decrease in q is proportional to $|\hat{\lambda}_j|$.
- The step along p may be short if blocked by a new constraint, so the decrease in q is not guaranteed to be maximal.

Comments

Taking the inner product of both sides of the equation with p and using the fact that $a_i^T p = 0$ for $i \in \hat{\mathcal{W}}$ excluding j gives $-\hat{\lambda}_j a_j^T p = p^T G p$. Because $p^T G p$ is nonnegative and $\hat{\lambda}_j$ is negative, it follows that $a_j^T p \geq 0$. This confirms that p points in a feasible direction for constraint j.

If second-order sufficient conditions hold, $Z^T G Z$ is positive definite. Then $a_j^T p = 0$ only if $p^T G p = 0$, which implies that p itself is zero. But if $p = 0$, linear independence of the a_i implies $\hat{\lambda}_j$ must be zero, contradicting the assumption that it is negative. Hence, $p^T G p > 0$, and $a_j^T p > 0$ as well.

In practice, when several multipliers are negative, the most negative $\hat{\lambda}_j$ is often chosen. Sensitivity analysis shows that the decrease in the quadratic objective q is approximately proportional to the magnitude of $\hat{\lambda}_j$. The step along p may be shortened if blocked by other constraints, so the actual decrease in q may not reach its maximum possible value. Nevertheless, the direction is guaranteed to be feasible and provides systematic progress toward the optimum.

Consider the subproblem:

$$\min_{p} \ \frac{1}{2}p^{T}Gp + (Gx_k + c)^{T}p, \ \text{s.t.}\ a_i^{T}p = 0, \ \forall i \in \mathcal{W}_k, \qquad (*)$$

Theorem 39

Suppose that the solution p_k of the subproblem $(*)$ is nonzero and satisfies the second-order sufficient conditions. Then $q(\cdot)$ is strictly decreasing along p_k.

Proof:

Since p_k satisfies second-order conditions, $Z^{T}GZ$ is positive definite, where Z is a basis of the null space of $\left[a_i^{T}\right]_{i\in\mathcal{W}_k}$.

- Thus, p_k is the unique global solution of the subproblem.
- Since $p = 0$ is feasible, its objective value $\frac{1}{2}p^{T}Gp + (Gx_k + c)^{T}p$ at $p = 0$ is larger than at p_k, so:
$$\frac{1}{2}p_k^{T}Gp_k + (Gx_k + c)^{T}p_k < 0.$$
- Since $p_k^{T}Gp_k \geq 0$, we have $(Gx_k + c)^{T}p_k < 0$, so:
$$q(x_k + \alpha_k p_k) = q(x_k) + \alpha_k(Gx_k + c)^{T}p_k + \frac{1}{2}\alpha_k^2 p_k^{T}Gp_k < q(x_k),$$
for all $\alpha_k > 0$ sufficiently small. □

Comments

A key property of the subproblem solution p_k is that it produces a descent direction for the quadratic objective q when it is nonzero and satisfies second-order sufficient conditions. The subproblem minimizes $\frac{1}{2}p^{T}Gp + (Gx_k + c)^{T}p$, subject to $a_i^{T}p = 0$ for all i in the working set $\mathcal{W}_k$. Let Z denote a basis for the null space of the matrix formed by a_i^{T}. Second-order sufficient conditions ensure that $Z^{T}GZ$ is positive definite.

Since p_k lies in the column space of Z, it is the unique minimizer of the subproblem. Comparing the objective at $p = 0$, which is feasible, we see that the value at p_k is strictly smaller. This implies that the inner product of $Gx_k + c$ with p_k is negative. Therefore, for any sufficiently small positive α_k, $q(x_k + \alpha_k p_k) = q(x_k) + \alpha_k(Gx_k + c)^{T}p_k + \frac{1}{2}\alpha_k^2 p_k^{T}Gp_k$, which is strictly less than $q(x_k)$.

This proves that p_k is a descent direction for the quadratic function. The combination of feasible steps and guaranteed descent is crucial for the convergence of the active-set method. It ensures that each iteration either reduces the objective or adjusts the working set in a well-defined manner, systematically moving toward the optimal solution while maintaining feasibility.

Algorithm: Active-Set Method for Convex QP

Compute a feasible starting point x_0;
Set $\mathcal{W}_0$ to a subset of active constraints at x_0;
for $k = 0, 1, 2, \ldots$ do
  Solve $(*)$ to find p_k;
  if $p_k = 0$ then compute $\hat{\lambda}_i$ satisfying $Gx_k + c = \sum_{i \in \mathcal{W}_k} \hat{\lambda}_i a_i$;
    if $\hat{\lambda}_i \geq 0$ for all $i \in \mathcal{W}_k \cap \mathcal{I}$ then stop with solution $x^* = x_k$;
    else
      $j \leftarrow \arg\min_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j$;
      $x_{k+1} \leftarrow x_k$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$;
  else ($p_k \neq 0$)
    Compute step length α_k;
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
    if there are blocking constraints then
      $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{\text{one blocking constraint}\}$;
    else
      $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$;
      end (for)

Comments

The primal active-set algorithm for convex quadratic programming begins by computing a feasible starting point x_0 and defining an initial working set $\mathcal{W}_0$, typically a subset of active constraints at x_0. At each iteration, the subproblem is solved to find a step p_k. If $p_k = 0$, the Lagrange multipliers $\hat{\lambda}_i$ corresponding to constraints in $\mathcal{W}_k$ are computed from $Gx_k + c = \sum_{i \in \mathcal{W}_k} \hat{\lambda}_i a_i$.

If all $\hat{\lambda}_i$ for $i \in \mathcal{W}_k \cap \mathcal{I}$ are nonnegative, the current point satisfies all KKT conditions and the algorithm terminates with $x^* = x_k$. If some $\hat{\lambda}_j$ is negative, the corresponding constraint is removed from $\mathcal{W}_k$, and the next iteration begins with the same x_k but a reduced working set.

If p_k is nonzero, a step length α_k is computed to maintain feasibility, and the next iterate x_{k+1} is $x_k + \alpha_k p_k$. If any new blocking constraints are encountered along this step, one of them is added to $\mathcal{W}_{k+1}$; otherwise, the working set remains unchanged. Iterations continue in this manner until convergence.

This algorithm systematically alternates between adjusting the working set and computing feasible steps. The combination of step computation, Lagrange multiplier checks, and working set updates ensures that the method progresses toward a globally optimal solution, while maintaining feasibility and satisfying all KKT conditions. Its design guarantees convergence for convex quadratic programs, and it provides a clear operational framework for implementation.

Phase 1: Initial Feasible Point

Given estimate $\tilde{x}$, solve the feasibility LP:

$$\begin{aligned} \min_{\{x,z\}} \quad & e^T z, \\ \text{s.t.} \quad & a_i^T x + \gamma_i z_i = b_i, \quad i \in \mathcal{E}, \\ & a_i^T x + \gamma_i z_i \geq b_i, \quad i \in \mathcal{I}, \\ & z \geq 0, \end{aligned}$$

where $e = (1, 1, \ldots, 1)^T$, $\gamma_i = -\operatorname{sign}(a_i^T \tilde{x} - b_i)$ for $i \in \mathcal{E}$, $\gamma_i = 1$ for $i \in \mathcal{I}$.

Initial point:

$$x = \tilde{x}, \quad z_i = |a_i^T \tilde{x} - b_i| \ (i \in \mathcal{E}), \quad z_i = \max(b_i - a_i^T \tilde{x}, 0) \ (i \in \mathcal{I}).$$

- If $\tilde{x}$ is feasible for the QP $\min_x \frac{1}{2} x^T G x + c^T x$ s.t. $a_i^T x = b_i$, $i \in \mathcal{E}$, $a_i^T x \geq b_i$, $i \in \mathcal{I}$, then $(\tilde{x}, 0)$ is optimal.
- If the QP has feasible points, the LP's optimal value is zero, yielding a feasible x.
- $\mathcal{W}_0$ for Active-Set Method: linearly independent subset of active constraints at the LP solution.

Comments

When applying the primal active-set method, the first step is always to ensure that we have a feasible starting point. This is handled through what is called a phase one procedure. The idea is to take an initial estimate, denoted $\tilde{x}$, and to construct an auxiliary linear program that searches for feasibility. In this program we introduce slack variables, denoted z, which measure constraint violation. The linear objective minimizes the sum of all these violations, written as $e^T z$, where e is the vector of all ones. Each equality constraint is expressed as $a_i^T x + \gamma_i z_i = b_i$, and each inequality as $a_i^T x + \gamma_i z_i \geq b_i$, with $z \geq 0$. The coefficients γ_i are chosen to adjust the direction in which violation is measured: for equality constraints, $\gamma_i = -\operatorname{sign}(a_i^T \tilde{x} - b_i)$, while for inequalities $\gamma_i = 1$.

If the initial estimate $\tilde{x}$ already satisfies the quadratic program, then $(\tilde{x}, 0)$ is optimal for this feasibility problem. If feasible points exist at all, the optimal value of this linear program will be zero, giving us a feasible x to start the quadratic method. The first working set $\mathcal{W}_0$ is then chosen as a linearly independent subset of the constraints active at this feasible solution. This systematic approach guarantees that our algorithm always begins from a valid point.

"Big M" Penalty Method includes infeasibility measure η in the QP:

$$\begin{aligned} \min_{\{x,\eta\}} \quad & \frac{1}{2}x^T G x + x^T c + M\eta, \\ \text{s.t.} \quad & |(a_i^T x - b_i)| \le \eta, \quad i \in \mathcal{E}, \\ & b_i - a_i^T x \le \eta, \quad i \in \mathcal{I}, \\ & 0 \le \eta, \end{aligned}$$

for some large $M > 0$. As before, $\tilde{x}$ is the user-supplied initial guess.

- For feasible QP $\min_x \frac{1}{2}x^T G x + c^T x$ s.t. $a_i^T x = b_i$, $i \in \mathcal{E}$, $a_i^T x \ge b_i$, $i \in \mathcal{I}$, and large M, the solution has $\eta = 0$, with x solving the QP.
- Strategy: Choose M, solve the problem; if $\eta > 0$, increase M and retry.
- ℓ_1-norm variant: $\min_{\{x,s,t,v\}} \quad \frac{1}{2}x^T G x + x^T c + M e_{\mathcal{E}}^T (s+t) + M e_{\mathcal{I}}^T v,$

$$\begin{aligned} \text{s.t.} \quad & a_i^T x - b_i + s_i - t_i = 0, \quad i \in \mathcal{E}, \\ & a_i^T x - b_i + v_i \ge 0, \quad i \in \mathcal{I}, \\ & s \ge 0,\ t \ge 0,\ v \ge 0, \end{aligned}$$

$e_{\mathcal{E}}$ is the vector $\langle 1, 1, \ldots, 1\rangle^T$ of length $|\mathcal{E}|$; similarly for $e_{\mathcal{I}}$.

Comments

An alternative approach is the so-called big M penalty method. The principle here is to modify the quadratic program itself so that infeasibility is penalized directly in the objective. We introduce a nonnegative variable η, which measures the maximum violation across all constraints. The modified problem minimizes $\frac{1}{2}x^T G x + c^T x + M\eta$, subject to the condition that for each equality constraint the absolute value of $a_i^T x - b_i$ is less or equal to η, and for each inequality $b_i - a_i^T x$ is less or equal to η. The parameter M is a large positive constant, chosen so that feasibility is heavily favored. If the original quadratic program is feasible, the optimal solution of this penalized problem has $\eta = 0$ and gives exactly the same x as the feasible quadratic program. If $\eta > 0$, one can increase the value of M and resolve until feasibility dominates.

A common variant is based on the one-norm of the violations, which replaces the single η with separate nonnegative variables. For equalities, violations are split into positive and negative parts, denoted s and t, so that $a_i^T x - b_i + s_i - t_i = 0$. For inequalities, slack variables v are added. This approach provides finer control and avoids masking different sources of infeasibility under a single η.

Primal Active-Set Methods: Example

In the following example subscripts on x and p denote components (e.g., x_1), superscripts denote iteration (e.g., x^4).

Example

Apply Active-Set Method to the problem:

$$\begin{aligned} \min_{x} \quad & q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2, \\ \text{s.t.} \quad & x_1 - 2x_2 + 2 \geq 0, \\ & -x_1 - 2x_2 + 6 \geq 0, \\ & -x_1 + 2x_2 + 2 \geq 0, \\ & x_1 \geq 0, \\ & x_2 \geq 0. \end{aligned}$$

Constraints are indexed 1 to 5.

- Feasible initial point: $x^0 = (2, 0)^T$.
- Constraints 3 and 5 are active at x^0, set $\mathcal{W}_0 = \{3, 5\}$.
- Other choices (e.g., $\mathcal{W}_0 = \{5\}$, $\mathcal{W}_0 = \{3\}$, or $\mathcal{W}_0 = \emptyset$) are valid, affecting algorithm behavior.

Comments

Let's consider a simple quadratic program to see how the primal active-set method works in practice. The objective function is $(x_1-1)^2+(x_2-2.5)^2$. Without constraints, the minimum would be at $x = (1, 2.5)^T$. But we impose five inequalities that form a polygonal feasible region in the first quadrant.

As the starting point, we take $x^0 = (2, 0)^T$. This point satisfies all constraints, so it is feasible. At this point, the third and the fifth constraints are active: the third because $-x_1 + 2x_2 + 2 = 0$, and the fifth because $x_2 = 0$. So, the initial working set is chosen as constraints three and five.

It is important to note that this choice is not unique. We could start with only one of these active, or even with none. But whichever option we choose, the algorithm will eventually find the same solution. What changes is only the path — the sequence of working sets and iterates the method goes through. This illustrates one of the strengths of the active-set framework: flexibility in the starting configuration while preserving global convergence.

At $x^0 = (2,0)^T$, $\mathcal{W}_0 = \{3,5\}$, x^0 is a vertex, so $p^0 = 0$ for:

$$\min_p \quad \frac{1}{2} p^T G p + (G x^0 + c)^T p,$$
$$\text{s.t.} \quad a_i^T p = 0, \quad i \in \mathcal{W}_0.$$

Multipliers from $G x^0 + c = \sum_{i \in \mathcal{W}_0} \hat{\lambda}_i a_i$ $\left(\text{here } G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix},\ c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}\right)$:

$$\begin{bmatrix} -1 \\ 2 \end{bmatrix} \hat{\lambda}_3 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \hat{\lambda}_5 = \begin{bmatrix} 2 \\ -5 \end{bmatrix}, \quad \langle \hat{\lambda}_3, \hat{\lambda}_5 \rangle = \langle -2, -1 \rangle.$$

- Remove constraint 3 ($\hat{\lambda}_3 = -2$), set $\mathcal{W}_1 = \{5\}$, $x^1 = (2,0)^T$.
- Iteration 1: Solve QP for $\mathcal{W}_1$, get $p^1 = (-1,0)^T$, step length $\alpha_1 = 1$, so $x^2 = (1,0)^T$. No blocking constraints, so $\mathcal{W}_2 = \mathcal{W}_1 = \{5\}$.
- Iteration 2: Solve QP, get $p^2 = 0$. Multiplier: $\hat{\lambda}_5 = -5$. Drop constraint 5, set $\mathcal{W}_3 = \emptyset$.

Comments

At the starting point x^0, we solve the quadratic subproblem restricted by the active constraints three and five. Because this point is already a vertex, and therefore the corresponding columns of matrix A are linearly independent, the step p^0, satisfying the constraints, is zero. To test whether the current point is optimal, we compute the Lagrange multipliers. These multipliers measure whether the active constraints support an optimal solution. If they are all nonnegative, we have optimality. But here both multipliers turn out negative. That means the current point cannot be optimal with both constraints active.

According to the rules of the active-set method, we remove one of the constraints with a negative multiplier. Let's drop the third constraint corresponding to the smallest multiplier. The working set now contains only constraint five, while the iterate itself remains unchanged.

With this new set, solving the subproblem gives a nonzero step: $p^1 = (-1,0)^T$. Taking this step moves us to $x^2 = (1,0)^T$. No new constraints become active, so the working set remains $\{5\}$. In the next subproblem, however, we again get zero step, but the multiplier for constraint five is negative. Therefore, we must drop constraint five as well. The working set becomes empty, which sets the stage for solving the unconstrained problem in the next iteration.

Objective: $q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2$, $G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$.

- Iteration 3: $\mathcal{W}_3 = \emptyset$, solve unconstrained QP: $\min_p \frac{1}{2} p^T G p + (Gx^3 + c)^T p$, $x^3 = (1, 0)^T$.
  - $Gx^3 + c = \begin{bmatrix} 2 \cdot 1 - 2 \\ 2 \cdot 0 - 5 \end{bmatrix} = \begin{bmatrix} 0 \\ -5 \end{bmatrix}$, solution $p^3 = (0, 2.5)^T$.
  - For $x^4 = x^3 + \alpha p^3 = (1, \alpha \cdot 2.5)$, substitute into constraints:

$$
\begin{aligned}
&1: x_1 - 2x_2 + 2 \geq 0 \;\Rightarrow\; 1 - 2 \cdot (\alpha \cdot 2.5) + 2 = 3 - 5\alpha \geq 0 \Rightarrow \alpha \leq 0.6, \\
&2: -x_1 - 2x_2 + 6 \geq 0 \;\Rightarrow\; -1 - 2 \cdot (\alpha \cdot 2.5) + 6 = 5 - 5\alpha \geq 0 \Rightarrow \alpha \leq 1.0, \\
&3: -x_1 + 2x_2 + 2 \geq 0 \;\Rightarrow\; -1 + 2 \cdot (\alpha \cdot 2.5) + 2 = 1 + 5\alpha \geq 0 \Rightarrow \alpha \geq -0.2, \\
&4: x_1 \geq 0 \;\Rightarrow\; x_1 = 1 \geq 0, \text{ satisfied}, \\
&5: x_2 \geq 0 \;\Rightarrow\; x_2 = \alpha \cdot 2.5 \geq 0, \text{ satisfied for } \alpha \geq 0.
\end{aligned}
$$

  - $\alpha_3 = \min(0.6, 1.0) = 0.6$ (constraint 1 active), so $x^4 = (1, 1.5)^T$, $\mathcal{W}_4 = \{1\}$.

Comments

With no active constraints, the method reduces to solving the unconstrained quadratic problem with respect to p. The step we obtain is $p^3 = (0, 2.5)^T$. So the algorithm wants to move straight upward.

Before taking this step fully, we must check feasibility. Substituting the candidate $x = (1, \alpha \cdot 2.5)$ into the constraints, we find the tightest restriction comes from the first inequality, which requires $\alpha \leq 0.6$. That means we cannot move all the way to the unconstrained minimizer; instead, we stop earlier, exactly when constraint one becomes active.

Thus, the step length is $\alpha_3 = 0.6$. The new point is $x^4 = (1, 1.5)^T$. At this location, constraint one is active, so the working set for the next iteration is $\{1\}$. This shows how the algorithm balances between moving in a descent direction and respecting feasibility by stopping at the boundary of the feasible set.

Objective: $q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2$, $G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $c = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$.

Iteration 4: Solve QP for $\mathcal{W}_4 = \{1\}$, $a_1 = (1, -2)^T$: $p_1 - 2p_2 = 0$.

- $x^4 = (1, 1.5)^T$, $Gx^4 + c = \begin{bmatrix} 2 \cdot 1 - 2 \\ 2 \cdot 1.5 - 5 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.
- QP: $\min_{p_1} 5p_2^2 - 2p_2$ s.t. $p_1 = 2p_2$, solution $p^4 = (0.4, 0.2)^T$, step length $\alpha_4 = 1$, so $x^5 = (1 + 0.4, 1.5 + 0.2) = (1.4, 1.7)^T$.
- No blocking constraints, so $\mathcal{W}_5 = \mathcal{W}_4 = \{1\}$.

Iteration 5: Solve QP for $\mathcal{W}_5 = \{1\}$, get $p^5 = (0, 0)^T$.

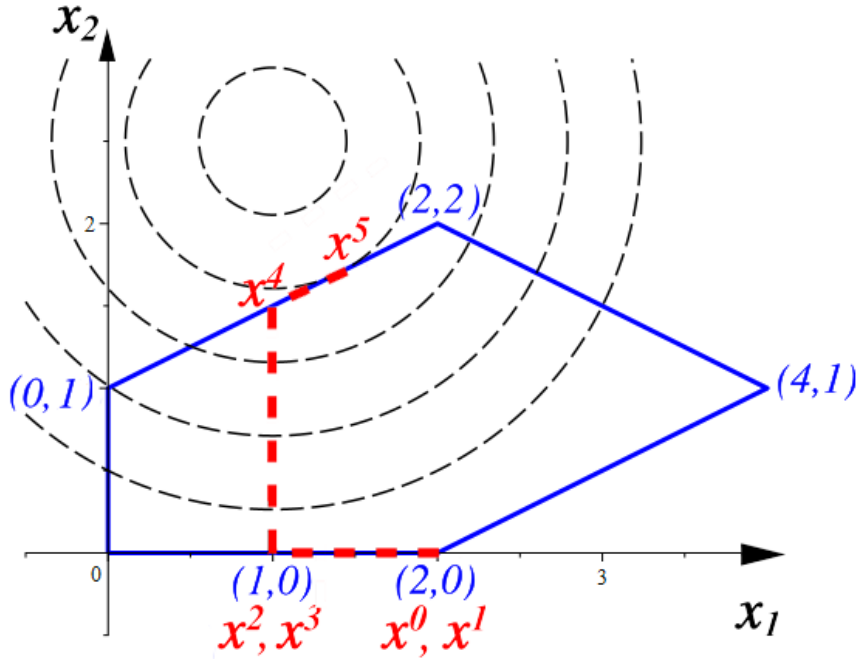
- Multiplier: $Gx^5 + c = \begin{bmatrix} 2 \cdot 1.4 - 2 \\ 2 \cdot 1.7 - 5 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -1.6 \end{bmatrix} = \hat{\lambda}_1 \begin{bmatrix} 1 \\ -2 \end{bmatrix}$, so $\hat{\lambda}_1 = 0.8$.
- Since $\hat{\lambda}_1 \geq 0$, stop with $x^* = (1.4, 1.7)^T$.

Comments

At iteration four, the working set contains only constraint one. This means our step must remain tangent to the boundary defined by $x_1 - 2x_2 + 2 = 0$. Solving the subproblem gives a direction $p^4 = (0.4, 0.2)^T$. Taking a full step moves us to $x^5 = (1.4, 1.7)^T$. No new constraints become active, so the working set stays the same.

In iteration five, solving again produces zero step. That is a signal that we might have reached optimality. To confirm, we compute the multiplier associated with the active constraint. It comes out positive. This tells us the Karush-Kuhn-Tucker conditions are satisfied: the point is feasible, the gradient is balanced by the active constraint, and the multiplier is nonnegative. Therefore, the algorithm stops here.

The final solution is $x^* = (1.4, 1.7)^T$. The example shows clearly how the active-set method proceeds: starting from a feasible point, it alternates between moving along descent directions and adjusting the set of active constraints, until the conditions for optimality are fully met.



Comments

This figure gives us a geometric view of how the active-set method behaves on the quadratic example we have just solved. The dashed black curves are the level sets of the objective function. They show where the function has the same value, and their centers mark the unconstrained minimizer at point $(1, 2.5)^{\mathrm{T}}$. The solid blue curve outlines the feasible region defined by the inequalities.

The red polyline is the trajectory of the algorithm. We start at the point $(2, 0)^{\mathrm{T}}$, lying on the boundary. From there, the method takes steps along edges of the feasible set or moves inside, depending on the current working set. Each kink in the red path corresponds to a change in the set of active constraints: either dropping one with a negative multiplier or adding a new one when we hit the boundary.

Notice that the path does not move directly toward the unconstrained minimizer, because feasibility must always be maintained. Instead, the algorithm zigzags between descent directions and boundary adjustments. Finally, it settles at the point $(1.4, 1.7)^{\mathrm{T}}$, where the gradient is balanced against the active constraint and all optimality conditions are satisfied.

This picture nicely summarizes the logic of the active-set approach: optimization proceeds as a sequence of constrained steps, adapting the working set until the solution is reached.

Choice of Initial Working Set

In the previous example for $x^0 = (2,0)^T$, active constraints were $\mathcal{W}_0 = \{3,5\}$. Alternatives:

- $\mathcal{W}_0 = \{3\}$: $p^0 = (0.2, 0.1)^T$, $x^1 = (2.2, 0.1)^T$.
- $\mathcal{W}_0 = \{5\}$: $p^0 = (-1,0)^T$, $x^1 = (1,0)^T$ (skips dropping constraint 3).
- $\mathcal{W}_0 = \emptyset$: $p^1 = (-1, 2.5)^T$, $\alpha_1 = \frac{2}{3}$, $x^1 = \left(\frac{4}{3}, \frac{5}{3}\right)^T$, $\mathcal{W}_1 = \{1\}$, solution x^* on next iteration.

$\mathcal{W}_k$ and $\mathcal{A}(x^k)$ may differ later:

- If multiple blocking constraints, only one added to $\mathcal{W}_k$, breaking $\mathcal{W}_k = \mathcal{A}(x^k)$
- Choice of blocking constraint affects subsequent iterates.

Properties of Active-Set Method

Linear Independence: Constraint gradients in $\mathcal{W}_0$ are linearly independent. Strategy ensures this for all $\mathcal{W}_k$:

- Blocking constraint's normal a_i is not a linear combination of $\{a_i \mid i \in \mathcal{W}_k\}$ (see properties of blocking constraints).
- Deletion of constraints preserves linear independence.

Comments

When we begin the active-set method, the choice of the initial working set plays an important role in determining the sequence of iterates, although it does not affect the final solution. For instance, in the example with starting point $x^0 = (2,0)^T$, we could select constraints three and five as the working set, which was the choice we discussed earlier. But alternative choices are possible. If we begin with only constraint three, then the computed step is $p^0 = (0.2, 0.1)^T$, leading to $x^1 = (2.2, 0.1)^T$. If instead we select only constraint five, the step is $p^0 = (-1,0)^T$, giving $x^1 = (1,0)^T$, thus skipping the process of dropping constraint three later. Finally, if we start with an empty working set, the first step is $p^1 = (-1, 2.5)^T$, with step length $\alpha = 2/3$. This produces $x^1 = (4/3, 5/3)^T$, and in the next iteration constraint one becomes active.

This example highlights two important facts. First, the working set at iteration k does not always coincide with the set of constraints active at x^k. Second, linear independence of constraint gradients is preserved throughout. Each added blocking constraint has a normal vector that is linearly independent of the current set, and deletion of constraints never introduces dependence. These structural properties guarantee that the algorithm remains well-defined at every step.

Constraint Removal: Remove constraint with most negative Lagrange multiplier $\hat{\lambda}_i$:

- Effective but sensitive to constraint scaling (scaling constraint by $\beta > 0$ scales λ_i by $1/\beta$).
- Analogous to Dantzig's pivot rule in simplex method.

Iteration Bound: Adding/removing at most one constraint per iteration implies $\geq m$ iterations if m constraints are active at x^* and x^0 is strictly feasible.

Convergence for Strictly Convex QPs

Active-Set Method for QP: $\min_x \frac{1}{2}x^T Gx + c^T x$, s.t. $a_i^T x = b_i$, $i \in \mathcal{E}$, $a_i^T x \geq b_i$, $i \in \mathcal{I}$, converges to x^* in finite iterations if $\alpha_k > 0$ when $p_k \neq 0$.

If $p_k = 0$ solves $\min_p \frac{1}{2}p^T Gp + (Gx_k + c)^T p$, s.t. $a_i^T p = 0$, $i \in \mathcal{W}_k$, then x_k is the unique global minimizer of $q(\cdot)$ for $\mathcal{W}_k$ (see descent direction properties).

- If $\hat{\lambda}_i < 0$ for some $i \in \mathcal{W}_k$, dropping i yields $p_{k+1} \neq 0$, a strict descent direction for $q(x)$.
- Since $\alpha_k > 0$, $q(x_{k+1}) < q(x_k)$, so $\mathcal{W}_k$ is never revisited.

Comments

Two central properties govern the behavior of the active-set method: the rule for removing constraints and the guarantee of convergence for strictly convex quadratic programs. When we encounter a zero step, we compute the Lagrange multipliers for the current working set. If one of them is negative, we must remove the corresponding constraint. The standard rule is to delete the constraint with the most negative multiplier. This choice is effective but has one drawback: it depends on scaling. If a constraint is multiplied by a positive factor β, then its multiplier scales by $1/\beta$, which can change which constraint looks "most negative." This sensitivity is very similar to what happens in the simplex method, where Dantzig's pivot rule selects the variable with the largest reduced cost.

Another property is that at most one constraint is added or removed per iteration. Suppose the optimal point has m active constraints, and we start from a strictly feasible x^0 with none of them active. Then, in the worst case, we need at least m iterations, since each step adds at most one constraint. Fortunately, for strictly convex quadratic programs, we can guarantee finite termination. Whenever p_k is nonzero, the step length α_k is strictly positive, so the objective decreases. If p_k is zero, but some multiplier is negative, removing that constraint produces a strict descent direction. Because the objective always decreases and no working set is ever repeated, the algorithm must terminate in finitely many steps at the unique global solution.

Convergence of Active-Set Method (Part 1)

If $p_k \neq 0$, either $\alpha_k = 1$ (reaching minimizer for $\mathcal{W}_k$, so $p_{k+1} = 0$) or a constraint is added to $\mathcal{W}_k$.

- After $\leq n$ iterations, $\mathcal{W}_k$ has n independent constraints, forcing $p_k = 0$.

Convergence and Cycling

Finite Termination: Since $p_k = 0$ occurs at least every n iterations and $\mathcal{W}_k$ is never revisited, the finite number of possible $\mathcal{W}_k$ ensures termination at x^* satisfying optimality for the QP.

Cycling: $\alpha_k > 0$ for $p_k \neq 0$ prevents cycling (where $x^k = x^{k+s}$, $\mathcal{W}_k = \mathcal{W}_{k+l}$ for some integers k and for some $s \geq 1$).

- Cycling occurs if constraints are dropped/added without any movement along the computed direction p.
- Handled similarly to linear programming methods; most QP implementations ignore cycling.

Comments

The convergence of the active-set method relies on two key mechanisms: the behavior of the search direction and the evolution of the working set. Whenever the computed step direction, denoted p_k, is nonzero, the method always makes progress. Either the step length α_k equals one, which means we have reached the minimizer relative to the current working set and so the next direction p_{k+1} becomes zero, or a new constraint becomes active and is added to the working set. Because the dimension of the problem is n, after at most n consecutive steps, we accumulate n linearly independent constraints, which forces the direction p_k to be zero. This ensures that the algorithm cannot continue indefinitely without reaching a stationary situation.

When $p_k = 0$, we check multipliers and either terminate at the optimal solution or drop a constraint. Thus, the situation $p_k = 0$ occurs at least once every n iterations, guaranteeing steady progress. Moreover, the working set is never revisited. Since the number of possible working sets is finite, the method must terminate at a point x^* satisfying the Karush-Kuhn-Tucker conditions.

Cycling, which would mean revisiting the same point with the same working set, is excluded by the fact that α_k is strictly positive whenever p_k is nonzero. Cycling could only occur if constraints were added and removed without any actual step. In practice, this is treated similarly to the simplex method, and most quadratic programming implementations safely ignore it.

Computing Steps in Active-Set Method

Step for QP: $\min_p \frac{1}{2}p^T Gp + (Gx_k + c)^T p$, s.t. $a_i^T p = 0$, $i \in \mathcal{W}_k$, solves KKT system:

$$\begin{bmatrix} G & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} -(Gx_k + c) \\ 0 \end{bmatrix}.$$

- $\mathcal{W}_k$ changes by ≤ 1 index per iteration, so KKT matrix differs by ≤ 1 row/column (G fixed, A changes).
- Update matrix factors from previous iteration instead of recomputing.

Null-space method: A has m independent rows, QR factorization $A^T\Pi = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$, where Π is a permutation matrix; R is square, upper triangular and nonsingular; $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ is $n \times n$ orthogonal; and Q_1 and R both have m columns while Q_2 has $n - m$ columns. We can choose $Z = Q_2$.

- New constraint: $\tilde{A}^T = \begin{bmatrix} A^T & a \end{bmatrix}$, where a is a column vector of length n such that $\tilde{A}^T$ retains full column rank. Let's update Q, R to get $\tilde{Z}$ (with $n - m - 1$ columns) for the expanded matrix $\tilde{A}$. Since $Q_1 Q_1^T + Q_2 Q_2^T = I$ we have:

$$\tilde{A}^T \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A^T\Pi & a \end{bmatrix} = Q \begin{bmatrix} R & Q_1^T a \\ 0 & Q_2^T a \end{bmatrix}.$$

Comments

The core computational task in the active-set method is the determination of the search direction. At iteration k, we solve a quadratic subproblem in the variable p: minimize $\frac{1}{2}p^T Gp + (Gx_k + c)^T p$, subject to the linear equalities $a_i^T p = 0$ for all i in the working set. This is equivalent to a Karush-Kuhn-Tucker system, written as a block matrix with G in the upper left, the active constraint matrix A in the upper right, A^T in the lower left, and zeros in the lower right. Solving this system yields both the step p and the multipliers λ.

Because the working set changes by at most one constraint per iteration, the KKT matrix also changes by at most one row and one column. This structure makes it possible to update matrix factorizations efficiently instead of recomputing them from scratch.

One popular approach is the null-space method. If A has m independent rows, we can compute a QR factorization of A^T with permutation, so that $A^T\Pi = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$. Here R is a nonsingular upper triangular matrix of size $m \times m$, and Q is orthogonal, partitioned as Q_1 and Q_2. The columns of Q_2 form a basis of the null space, which we denote as Z. When a new constraint a is added, we extend A^T to a new matrix and update the factorization to obtain a new null-space basis. This update is more efficient than starting over, and the orthogonal structure ensures numerical stability.

QR Update for Adding Constraint

Add constraint to $\mathcal{W}_k$, new matrix $\tilde{A}^T = \begin{bmatrix} A^T & a \end{bmatrix}$. Update QR factorization to get new null-space basis $\tilde{Z}$.

- Define orthogonal $\hat{Q}$ s.t.:

$$\hat{Q}(Q_2^T a) = \begin{bmatrix} \gamma \\ 0 \end{bmatrix}, \quad \|Q_2^T a\| = |\gamma|, \text{ where } \gamma \text{ is a scalar.}$$

- Update QR:

$$\tilde{A}^T \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} = Q \begin{bmatrix} R & Q_1^T a \\ 0 & \hat{Q}^T \begin{bmatrix} \gamma \\ 0 \end{bmatrix} \end{bmatrix} = Q \begin{bmatrix} I & 0 \\ 0 & \hat{Q}^T \end{bmatrix} \begin{bmatrix} R & Q_1^T a \\ 0 & \gamma \\ 0 & 0 \end{bmatrix} = \tilde{Q} \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix},$$

$$\tilde{\Pi} = \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix}, \quad \tilde{Q} = \begin{bmatrix} Q_1 & Q_2 \hat{Q}^T \end{bmatrix}, \quad \tilde{R} = \begin{bmatrix} R & Q_1^T a \\ 0 & \gamma \end{bmatrix}.$$

- $\tilde{Z}$ = last $n - m - 1$ columns of $Q_2 \hat{Q}^T$.
- Efficiency: Computing $\hat{Q}$, $Q_2 \hat{Q}^T$ costs $\mathcal{O}(n(n-m))$, vs. $\mathcal{O}(n^2 m)$ for full QR, especially efficient when $n - m \ll n$.

Comments

When a new constraint enters the working set, we must update the null-space basis accordingly. Suppose the new active matrix is A^T augmented with an additional column a. To avoid recomputing a full QR factorization, we apply a structured update. First, we compute an orthogonal transformation, denoted $\hat{Q}$, such that $\hat{Q}$ applied to $Q_2^T a$ produces a vector with γ in the first entry and zeros elsewhere, where γ is the norm of $Q_2^T a$. This isolates the new contribution in a single coordinate.

The update proceeds by combining the original Q with this transformation, forming a new orthogonal matrix $\tilde{Q}$, and a new upper triangular matrix $\tilde{R}$. From these, the expanded null-space basis $\tilde{Z}$ is taken as the last $n - m - 1$ columns of $Q_2 \hat{Q}^T$. The essential point is that the update cost is only of order $n(n-m)$, significantly cheaper than recomputing a full QR, which would cost on the order of $n^2 m$. This difference is especially important when $n - m$, the dimension of the null space, is small relative to n.

Thus, by cleverly updating factorizations, the active-set method maintains efficiency across iterations. The algorithm adapts to each new constraint with only incremental work, while preserving the orthogonality and stability properties crucial for accurate computations.

QR Update (Removing Constraint) and Hessian

- **Remove constraint**: Delete row from A, column from R. Restore upper triangularity in R via plane rotations.
- **Update** Q by inexpensive transformation of its first m columns, and the updated null-space matrix is the last $n - m + 1$ columns from this matrix after the transformations are complete:
$$\tilde{Z} = \begin{bmatrix} \tilde{z} & Z \end{bmatrix}.$$
- **Cost**: Cheaper than $\mathcal{O}(n^2 m)$ for full QR (see numerical linear algebra methods).
- **Reduced Hessian**: For QP, $h = 0$, $p_Y = 0$, solve:
$$(Z^T G Z) p_z = -Z^T (G x_k + c).$$
- **Update Cholesky**: $Z^T G Z = L L^T$. Then for $\tilde{Z} = \begin{bmatrix} \tilde{z} & Z \end{bmatrix}$, transform L to $\tilde{L}$ for $\tilde{Z}^T G \tilde{Z}$ via elementary operations.
- **Simplifications**: Update $Z^T(Gx_k + c)$ with Z to $\tilde{Z}$ (see further simplifications).

Comments

The reverse operation occurs when a constraint is removed from the working set. In this case, the active matrix loses a row, and correspondingly the triangular factor R loses a column. To restore the triangular form, we apply plane rotations, a standard tool in numerical linear algebra. The orthogonal matrix Q is then updated by inexpensive transformations of its first m columns, and the new null-space basis is obtained as the last $n - m + 1$ columns of the updated Q. Symbolically, the new $\tilde{Z}$ is constructed by adding one extra column vector to the previous basis.

The computational cost of this operation is much less than recomputing a full QR factorization, making constraint removal efficient as well. With both adding and dropping constraints handled incrementally, the method adapts dynamically while keeping computations light.

Once the null-space basis is updated, we must also update the reduced Hessian, defined as $Z^T G Z$. This reduced matrix governs the quadratic behavior in the feasible subspace. To solve for the search step, we form the equation $Z^T G Z p_z = -Z^T(Gx_k + c)$. The system is solved efficiently using a Cholesky factorization, denoted LL^T. When Z is replaced by $\tilde{Z}$, the factor L is transformed into a new $\tilde{L}$ using elementary operations, again without starting over.

This interplay between updating factorizations and maintaining reduced Hessians is essential. It ensures that each iteration of the active-set method remains computationally feasible while rigorously preserving the mathematical structure required for convergence.