Chapter 9

Constrained Optimization

We continue our consideration of optimization problems by studying the *constrained* case. These problems take the following general form:

$$\begin{aligned} &\text{minimize } f(\vec{x}) \\ &\text{such that } g(\vec{x}) = \vec{0} \\ &\qquad\qquad h(\vec{x}) \geq \vec{0} \end{aligned}$$

Here, $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, and $h : \mathbb{R}^n \to \mathbb{R}^p$. Obviously this form is extremely generic, so it is not difficult to predict that algorithms for solving such problems in the absence of additional assumptions on f, g, or h can be difficult to formulate and are subject to degeneracies such as local minima and lack of convergence. In fact, this optimization encodes other problems we already have considered; if we take $f(\vec{x}) \equiv 0$, then this constrained optimization becomes root-finding on g, while if we take $g(\vec{x}) = h(\vec{x}) \equiv \vec{0}$ then it reduces to unconstrained optimization on f.

Despite this somewhat bleak outlook, optimizations for general constrained case can be valuable when f, g, and h do not have useful structure or are too specialized to merit specialized treatment. Furthermore, when f is heuristic anyway, simply finding a feasible $\vec{x}$ for which $f(\vec{x}) < f(\vec{x}_0)$ for an initial guess $\vec{x}_0$ is valuable. One simple application in this domain would be an economic system in which f measures costs; obviously we wish to minimize costs, but if $\vec{x}_0$ represents the current configuration, *any* $\vec{x}$ decreasing f is a valuable output.

9.1 Motivation

It is not difficult to encounter constrained optimization problems in practice. In fact, we already listed many applications of these problems when we discussed eigenvectors and eigenvalues, since this problem can be posed as finding critical points of $\vec{x}^\top A\vec{x}$ subject to $\|\vec{x}\|_2 = 1$; of course, the particular case of eigenvalue computation admits special algorithms that make it a simpler problem.

Here we list other optimizations that do not enjoy the structure of eigenvalue problems:

Example 9.1 (Geometric projection)**.** *Many surfaces S in $\mathbb{R}^3$ can be written* implicitly *in the form $g(\vec{x}) = 0$ for some g. For example, the unit sphere results from taking $g(\vec{x}) \equiv \|\vec{x}\|_2^2 - 1$, while a cube can*

be constructed by taking $g(\vec{x}) = \|\vec{x}\|_1 - 1$. In fact, some 3D modeling environments allow users to specify "blobby" objects, as in Figure NUMBER, as sums

$$g(\vec{x}) \equiv c + \sum_i a_i e^{-b_i \|\vec{x} - \vec{x}_i\|_2^2}.$$

Suppose we are given a point $\vec{y} \in \mathbb{R}^3$ and wish to find the closest point on S to $\vec{y}$. This problem is solved by using the following constrained minimization:

$$\begin{aligned} &\text{minimize}_{\vec{x}} \ \|\vec{x} - \vec{y}\|_2 \\ &\text{such that } g(\vec{x}) = 0 \end{aligned}$$

Example 9.2 (Manufacturing)**.** *Suppose you have m different materials; you have s_i units of each material i in stock. You can manufacture k different products; product j gives you profit p_j and uses c_{ij} of material i to make. To maximize profits, you can solve the following optimization for the total amount x_j you should manufacture of each item j:*

$$\begin{aligned} &\text{maximize}_{\vec{x}} \ \sum_{j=1}^{k} p_j x_j \\ &\text{such that } x_j \geq 0 \ \forall j \in \{1, \ldots, k\} \\ &\sum_{j=1}^{k} c_{ij} x_j \leq s_i \ \forall i \in \{1, \ldots, m\} \end{aligned}$$

The first constraint ensures that you do not make negative numbers of any product, and the second ensures that you do not use more than your stock of each material.

Example 9.3 (Nonnegative least-squares)**.** *We already have seen numerous examples of least-squares problems, but sometimes negative values in the solution vector might not make sense. For example, in computer graphics, an animated model might be expressed as a deforming bone structure plus a meshed "skin;" for each point on the skin a list of weights can be computed to approximate the influence of the positions of the bone joints on the position of the skin vertices (CITE). Such weights should be constrained to be nonnegative to avoid degenerate behavior while the surface deforms. In such a case, we can solve the "nonnegative least-squares" problem:*

$$\begin{aligned} &\text{minimize}_{\vec{x}} \ \|A\vec{x} - \vec{b}\|_2 \\ &\text{such that } x_i \geq 0 \ \forall i \end{aligned}$$

Recent research involves characterizing the sparsity *of nonnegative least squares solutions, which often have several values x_i satisfying $x_i = 0$ exactly (CITE).*

Example 9.4 (Bundle adjustment)**.** *In computer vision, suppose we take a picture of an object from several angles. A natural task is to reconstruct the three-dimensional shape of the object. To do so, we might mark a corresponding set of points on each image; in particular, we can take $\vec{x}_{ij} \in \mathbb{R}^2$ to be the position of feature point j on image i. In reality, each feature point has a position $\vec{y}_j \in \mathbb{R}^3$ in space, which we would like to compute. Additionally, we must find the positions of the cameras themselves, which we can represent as*

unknown projection matrices P_i. This problem, known as bundle adjustment*, can be approached using an optimization strategy:*

$$\begin{aligned} &\text{minimize}_{\vec{y}_j, P_i} \sum_{ij} \|P_i \vec{y}_j - \vec{x}_{ij}\|_2^2 \\ &\text{such that} P_i \text{ is orthogonal } \forall i \end{aligned}$$

The orthogonality constraint ensures that the camera transformations are reasonable.

9.2 Theory of Constrained Optimization

In our discussion, we will assume that f, g, and h are differentiable. Some methods exist that only make weak continuity or Lipschitz assumptions, but these techniques are quite specialized and require advanced analytical consideration.

Although we have not yet developed algorithms for general constrained optimization, we implicitly have made use of the *theory* of such problems in considering eigenvalue methods. Specifically, recall the method of Lagrange multipliers, introduced in Theorem 0.1. In this technique, critical points $f(\vec{x})$ subject to $g(\vec{x})$ are characterized as critical points of the unconstrained Lagrange multiplier function $\Lambda(\vec{x}, \vec{\lambda}) \equiv f(\vec{x}) - \vec{\lambda} \cdot \vec{g}(\vec{x})$ with respect to both $\vec{\lambda}$ and $\vec{x}$ simultaneously. This theorem allowed us to provide variational interpretations of eigenvalue problems; more generally, it gives an alternative (necessary but not sufficient) criterion for $\vec{x}$ to be a critical point of an *equality-constrained* optimization.

Simply finding an $\vec{x}$ satisfying the constraints, however, can be a considerable challenge. We can separate these issues by making a few definitions:

Definition 9.1 (Feasible point and feasible set)**.** *A* feasible point *of a constrained optimization problem is any point $\vec{x}$ satisfying $g(\vec{x}) = \vec{0}$ and $h(\vec{x}) \geq \vec{0}$. The* feasible set *is the set of all points $\vec{x}$ satisfying these constraints.*

Definition 9.2 (Critical point of constrained optimization)**.** *A critical point of a constrained optimization is one satisfying the constraints that also is a local maximum, minimum, or saddle point of f within the feasible set.*

Constrained optimizations are difficult because they simultaneously solve root-finding problems (the $g(\vec{x}) = \vec{0}$ constraint), satisfiability problems (the $h(\vec{x}) \geq \vec{0}$ constraint), and minimization (the function f). This aside, to push our differential techniques to complete generality, we must find a way to add inequality constraints to the Lagrange multiplier system. Suppose we have found the minimum of the optimization, denoted $\vec{x}^*$. For each inequality constraint $h_i(\vec{x}^*) \geq 0$, we have two options:

- $h_i(\vec{x}^*) = 0$: Such a constraint is *active*, likely indicating that if the constraint were removed the optimum might change.
- $h_i(\vec{x}^*) > 0$: Such a constraint is *inactive*, meaning in a neighborhood of $\vec{x}^*$ if we had removed this constraint we still would have reached the same minimum.

Of course, we do not know which constraints will be active or inactive at $\vec{x}^*$ until it is computed.

If all of our constraints were active, then we could change our $h(\vec{x}) \geq \vec{0}$ constraint to an equality without affecting the minimum. This might motivate studying the following Lagrange multiplier system:

$$\Lambda(\vec{x}, \vec{\lambda}, \vec{\mu}) \equiv f(\vec{x}) - \vec{\lambda} \cdot g(\vec{x}) - \vec{\mu} \cdot h(x)$$

We no longer can say that $\vec{x}^*$ is a critical point of Λ, however, because inactive constraints would remove terms above. Ignoring this (important!) issue for the time being, we could proceed blindly and ask for critical points of this new Λ with respect to $\vec{x}$, which satisfy the following:

$$\vec{0} = \nabla f(\vec{x}) - \sum_i \lambda_i \nabla g_i(\vec{x}) - \sum_j \mu_j \nabla h_j(\vec{x})$$

Here we have separated out the individual components of g and h and treated them as scalar functions to avoid complex notation.

A clever trick can extend this optimality condition to inequality-constrained systems. Notice that if we had taken $\mu_j = 0$ whenever h_j is inactive, then this removes the irrelevant terms from the optimality conditions. In other words, we can *add* a constraint on the Lagrange multipliers:

$$\mu_j h_j(\vec{x}) = 0.$$

With this constraint in place, we know that at least one of μ_j and $h_j(\vec{x})$ must be zero, and therefore our first-order optimality condition still holds!

So far, our construction has not distinguished between the constraint $h_j(\vec{x}) \geq 0$ and the constraint $h_j(\vec{x}) \leq 0$. If the constraint is inactive, it could have been dropped without affecting the outcome of the optimization locally, so we consider the case when the constraint is active. Intuitively,[1] in this case we expect there to be a way to decrease f by violating the constraint. Locally, the direction in which f decreases is $-\nabla f(\vec{x}^*)$ and the direction in which h_j decreases is $-\nabla h_j(\vec{x}^*)$. Thus, starting at $\vec{x}^*$ we can decrease f even more by violating the constraint $h_j(\vec{x}) \geq 0$ when $\nabla f(\vec{x}^*) \cdot \nabla h_j(\vec{x}^*) \geq 0$.

Of course, products of gradients of f and h_j are difficult to work with. However, recall that at $\vec{x}^*$ our first-order optimality condition tells us:

$$\nabla f(\vec{x}^*) = \sum_i \lambda_i^* \nabla g_i(\vec{x}^*) + \sum_{j \text{ active}} \mu_j^* \nabla h_j(\vec{x}^*)$$

The inactive μ_j values are zero and can be removed. In fact, we can remove the $g(\vec{x}) = 0$ constraints by adding inequality constraints $g(\vec{x}) \geq \vec{0}$ and $g(\vec{x}) \leq \vec{0}$ to h; this is a mathematical convenience for writing a proof rather than a numerically-wise maneuver. Then, taking dot products with ∇h_k for any fixed k shows:

$$\sum_{j \text{ active}} \mu_j^* \nabla h_j(\vec{x}^*) \cdot \nabla h_k(\vec{x}^*) = \nabla f(\vec{x}^*) \cdot \nabla h_k(\vec{x}^*) \geq 0$$

Vectorizing this expression shows $Dh(\vec{x}^*)Dh(\vec{x}^*)^\top \vec{\mu}^* \geq \vec{0}$. Since $Dh(\vec{x}^*)Dh(x^*)^\top$ is positive semidefinite, this implies $\vec{\mu}^* \geq \vec{0}$. Thus, the $\nabla f(\vec{x}^*) \cdot \nabla h_j(\vec{x}^*) \geq 0$ observation manifests itself simply by the fact that $\mu_j \geq 0$.

Our observations can be formalized to prove a first-order optimality condition for inequality-constrained optimizations:

[1] You should not consider our discussion a formal proof, since we are not considering many boundary cases.

Theorem 9.1 (Karush-Kuhn-Tucker (KKT) conditions)**.** *The vector $\vec{x}^* \in \mathbb{R}^n$ is a critical point for minimizing f subject to $g(\vec{x}) = \vec{0}$ and $h(\vec{x}) \geq \vec{0}$ when there exists $\vec{\lambda} \in \mathbb{R}^m$ and $\vec{\mu} \in \mathbb{R}^p$ such that:*

- $\vec{0} = \nabla f(\vec{x}^*) - \sum_i \lambda_i \nabla g_i(\vec{x}^*) - \sum_j \mu_j \nabla h_j(\vec{x}^*)$ *("stationarity")*
- $g(\vec{x}^*) = \vec{0}$ *and* $h(\vec{x}^*) \geq \vec{0}$ *("primal feasibility")*
- $\mu_j h_j(\vec{x}^*) = 0$ *for all j ("complementary slackness")*
- $\mu_j \geq 0$ *for all j ("dual feasibility")*

Notice that when h is removed this theorem reduces to the Lagrange multiplier criterion.

Example 9.5 (Simple optimization[2])**.** *Suppose we wish to solve*

$$\begin{aligned} &\text{maximize } xy \\ &\text{such that } x + y^2 \leq 2 \\ &x, y \geq 0 \end{aligned}$$

In this case we will have no λ's and three μ's. We take $f(x,y) = -xy$, $h_1(x,y) \equiv 2 - x - y^2$, $h_2(x,y) = x$, and $h_3(x,y) = y$. The KKT conditions are:

$$\begin{aligned} \text{Stationarity: } & 0 = -y + \mu_1 - \mu_2 \\ & 0 = -x + 2\mu_1 y - \mu_3 \\ \text{Primal feasibility: } & x + y^2 \leq 2 \\ & x, y \geq 0 \\ \text{Complementary slackness: } & \mu_1(2 - x - y^2) = 0 \\ & \mu_2 x = 0 \\ & \mu_3 y = 0 \\ \text{Dual feasibility: } & \mu_1, \mu_2, \mu_3 \geq 0 \end{aligned}$$

Example 9.6 (Linear programming)**.** *Consider the optimization:*

$$\begin{aligned} &\text{minimize}_{\vec{x}} \ \vec{b} \cdot \vec{x} \\ &\text{such that } A\vec{x} \geq \vec{c} \end{aligned}$$

Notice Example 9.2 can be written this way. The KKT conditions for this problem are:

$$\begin{aligned} \text{Stationarity: } & A^\top \vec{\mu} = \vec{b} \\ \text{Primal feasibility: } & A\vec{x} \geq \vec{c} \\ \text{Complementary slackness: } & \mu_i(\vec{a}_i \cdot \vec{x} - c_i) = 0 \ \forall i, \text{ where } \vec{a}_i^\top \text{ is row } i \text{ of } A \\ \text{Dual feasibility: } & \vec{\mu} \geq \vec{0} \end{aligned}$$

As with the Lagrange multipliers case, we cannot assume that any $\vec{x}^*$ satisfying the KKT conditions automatically minimizes f subject to the constraints, even locally. One way to check for local optimality is to examine the Hessian of f restricted to the subspace of $\mathbb{R}^n$ in which $\vec{x}$ can move without violating the constraints; if this "reduced" Hessian is positive definite then the optimization has reached a local minimum.

[2]From `http://www.math.ubc.ca/~israel/m340/kkt2.pdf`

9.3 Optimization Algorithms

A careful consideration of algorithms for constrained optimization is out of the scope of our discussion; thankfully many stable implementations exist of these techniques and much can be accomplished as a "client" of this software rather than rewriting it from scratch. Even so, it is useful to sketch some potential approaches to gain some intuition for how these libraries function.

9.3.1 Sequential Quadratic Programming (SQP)

Similar to BFGS and other methods we considered in our discussion of unconstrained optimization, one typical strategy for constrained optimization is to approximate f, g, and h with simpler functions, solve the approximated optimization, and iterate.

Suppose we have a guess $\vec{x}_k$ of the solution to the constrained optimization problem. We could apply a second-order Taylor expansion to f and first-order approximation to g and h to define a next iterate as the following:

$$\begin{aligned}\vec{x}_{k+1} \equiv \vec{x}_k + \arg\min_{\vec{d}} \; & \frac{1}{2}\vec{d}^\top H_f(\vec{x}_k)\vec{d} + \nabla f(\vec{x}_k)\cdot\vec{d} + f(\vec{x}_k) \\ \text{such that } & g_i(\vec{x}_k) + \nabla g_i(\vec{x}_k)\cdot\vec{d} = 0 \\ & h_i(\vec{x}_k) + \nabla h_i(\vec{x}_k)\cdot\vec{d} \geq 0\end{aligned}$$

The optimization to find $\vec{d}$ has a quadratic objective with linear constraints, for which optimization can be considerably easier using one of many strategies. It is known as a *quadratic program*.

Of course, this Taylor approximation only works in a neighborhood of the optimal point. When a good initial guess $\vec{x}_0$ is unavailable, these strategies likely will fail.

Equality constraints When the only constraints are equalities and h is removed, the quadratic program for $\vec{d}$ has Lagrange multiplier optimality conditions derived as follows:

$$\begin{aligned}\Lambda(\vec{d},\vec{\lambda}) &\equiv \frac{1}{2}\vec{d}^\top H_f(\vec{x}_k)\vec{d} + \nabla f(\vec{x}_k)\cdot\vec{d} + f(\vec{x}_k) + \vec{\lambda}^\top(g(\vec{x}_k) + Dg(\vec{x}_k)\vec{d}) \\ \implies \vec{0} = \nabla_{\vec{d}}\Lambda &= H_f(\vec{x}_k)\vec{d} + \nabla f(\vec{x}_k) + [Dg(\vec{x}_k)]^\top\vec{\lambda}\end{aligned}$$

Combining this with the equality condition yields a linear system:

$$\begin{pmatrix} H_f(\vec{x}_k) & [Dg(\vec{x}_k)]^\top \\ Dg(\vec{x}_k) & 0 \end{pmatrix}\begin{pmatrix}\vec{d} \\ \vec{\lambda}\end{pmatrix} = \begin{pmatrix} -\nabla f(\vec{x}_k) \\ -g(\vec{x}_k)\end{pmatrix}$$

Thus, each iteration of sequential quadratic programming in the presence of only equality constraints can be accomplished by solving this linear system at each iteration to get $\vec{x}_{k+1} \equiv \vec{x}_k + \vec{d}$. It is important to note that the linear system above is *not* positive definite, so on a large scale it can be difficult to solve.

Extensions of this strategy operate as BFGS and similar approximations work for unconstrained optimization, by introducing approximations of the Hessian H_f. Stability also can be introduced by limiting the distance traveled in a single iteration.

Inequality Constraints Specialized algorithms exist for solving quadratic programs rather than general nonlinear programs, and these can be used to generate steps of SQP. One notable strategy is to keep an "active set" of constraints that are active at the minimum with respect to $\vec{d}$; then the equality-constrained methods above can be applied by ignoring inactive constraints. Iterations of active-set optimization update the active set of constraints by adding violated constraints to the active set and removing those inequality constraints h_j for which $\nabla f \cdot \nabla h_j \leq 0$ as in our discussion of the KKT conditions.

9.3.2 Barrier Methods

Another option for minimizing in the presence of constraints is to change the constraints to energy terms. For example, in the equality constrained case we could minimize an "augmented" objective as follows:

$$f_\rho(\vec{x}) = f(\vec{x}) + \rho \|g(\vec{x})\|_2^2$$

Notice that taking $\rho \to \infty$ will force $g(\vec{x})$ to be as small as possible, so eventually we will reach $g(\vec{x}) \approx \vec{0}$. Thus, the *barrier method* of constrained optimization applies iterative *unconstrained* optimization techniques to f_ρ and checks how well the constraints are satisfied; if they are not within a given tolerance, ρ is increased and the optimization continues using the previous iterate as a starting point.

Barrier methods are simple to implement and use, but they can exhibit some pernicious failure modes. In particular, as ρ increases, the influence of f on the objective function diminishes and the Hessian of f_ρ becomes more and more poorly conditioned.

Barrier methods can be applied to inequality constraints as well. Here we must ensure that $h_i(\vec{x}) \geq 0$ for all i; typical choices of barrier functions might include $1/h_i(\vec{x})$ (the "inverse barrier") or $-\log h_i(\vec{x})$ (the "logarithmic barrier").

9.4 Convex Programming

Generally speaking, methods like the ones we have described for constrained optimization come with few if any guarantees on the quality of the output. Certainly these methods are unable to obtain global minima without a good initial guess $\vec{x}_0$, and in certain cases, e.g. when the Hessian near $\vec{x}^*$ is not positive definite, they may not converge at all.

There is one notable exception to this rule, which appears in any number of important optimizations: convex programming. The idea here is that when f is a convex function and the feasible set itself is convex, then the optimization possesses a unique minimum. We already have defined a convex function, but need to understand what it means for a set of constraints to be convex:

Definition 9.3 (Convex set)**.** *A set $S \subseteq \mathbb{R}^n$ is* convex *if for any $\vec{x}, \vec{y} \in S$, the point $t\vec{x} + (1-t)\vec{y}$ is also in S for any $t \in [0,1]$.*

As shown in Figure NUMBER, intuitively a set is convex if its boundary shape cannot bend both inward and outward.

Example 9.7 (Circles)**.** *The disc $\{\vec{x} \in \mathbb{R}^n : \|\vec{x}\|_2 \leq 1\}$ is convex, while the unit circle $\{\vec{x} \in \mathbb{R}^n : \|\vec{x}\|_2 = 1\}$ is not.*

It is easy to see that a convex function has a unique minimum even when that function is restricted to a convex domain. In particular, if the function had two local minima, then the line of points between those minima must yield values of f not greater than those on the endpoints.

Strong convergence guarantees are available for convex optimizations that guarantee finding the global minimum so long as f is convex and the constraints on g and h make a convex feasible set. Thus, a valuable exercise for nearly any optimization problem is to check if it is convex, since such an observation can increase confidence in the solution quality and the chances of success by a large factor.

A new field called *disciplined convex programming* attempts to chain together simple rules about convexity to generate convex optimizations (CITE CVX), allowing the end user to combine simple convex energy terms and constraints so long as they satisfy criteria making the final optimization convex. Useful statements about convexity in this domain include the following:

- The intersection of convex sets is convex; thus, adding multiple convex constraints is an allowable operation.
- The sum of convex functions is convex.
- If f and g are convex, so is $h(\vec{x}) \equiv \max\{f(\vec{x}), g(\vec{x})\}$.
- If f is a convex function, the set $\{\vec{x} : f(\vec{x}) \leq c\}$ is convex.

Tools such as the `CVX` library help separate implementation of assorted convex objectives from their minimization.

Example 9.8 (Convex programming)**.**

- *The nonnegative least squares problem in Example 9.3 is convex because $\|A\vec{x} - \vec{b}\|_2$ is a convex function of $\vec{x}$ and the set $\vec{x} \geq \vec{0}$ is convex.*
- *The linear programming problem in Example 9.6 is convex because it has a linear objective and linear constraints.*
- *We can include $\|\vec{x}\|_1$ in a convex optimization objective by introducing a variable $\vec{y}$. To do so, we add constraints $y_i \geq x_i$ and $y_i \geq -x_i$ for each i and an objective $\sum_i y_i$. This sum has terms that are at least as large as $|x_i|$ and that the energy and constraints are convex. At the minimum we must have $y_i = |x_i|$ since we have constrained $y_i \geq |x_i|$ and we wish to minimize the energy. "Disciplined" convex libraries can do such operations behind the scenes without revealing such substitutions to the end user.*

A particularly important example of a convex optimization is *linear programing* from Example 9.6. The famous *simplex algorithm* keeps track of the active constraints, solves for the resulting $\vec{x}^*$ using a linear system, and checks if the active set must be updated; no Taylor approximations are needed because the objective and feasible set are given by linear machinery. *Interior point* linear programming strategies such as the barrier method also are successful for these problems. For this reason, linear programs can be solved on a *huge* scale—up to millions or billions of variables!—and often appear in problems like scheduling or pricing.

9.5 Problems

- Derive simplex?
- Linear programming duality