Optimization (an overview)

MTH 503

The *optimization problem* is to find a maximum or minimum of a function $f(x)$ subject to the constraint that $x \in \Omega$. Here $x \in \mathbf{R}^n$, and $\Omega \subset \mathbf{R}^n$ is the feasible region. That is, find a point $\bar{x} \in \Omega$ such that $f(\bar{x}) \geq f(x)$ or $f(\bar{x}) \leq f(x)$ for all $x \in \Omega$ (this $\bar{x}$ would be a *global* max or min, sometimes one may just look for a *local* max or min; $f(\bar{x}) \geq f(x)$ or $f(\bar{x}) \leq f(x)$ for all x nearby $\bar{x}$).

Linear Programming:

$$\begin{array}{rccl} \min & z & = & c \cdot x \\ \text{s.t.} & Ax & \leq & b \\ & x & \geq & 0 \end{array}$$

Here the objective function $f(x) = z = c \cdot x$ and the constraints $Ax \geq b,\ x \geq 0$, are linear (A is an $m \times n$ matrix), and $\Omega = \{x \in \mathbf{R}^n \mid Ax \leq b,\ x \geq 0\}$.

Simplex Method This algorithm (invented by Dantzig in 1947) uses the fact that the feasible region is a polyhedral set and the objective function is linear. The simplex method works very well in most cases and leads to many deep insights into the structure of linear programming problems (eg., duality). However, theoretically, the simplex method is an exponential-time algorithm.

Certain types of linear programming problems include networks (maximum flow, shortest path, etc), transportation problems, and integer programming (where one or more of the variables $x_1, x_2, \cdots, x_n$ are integers).

Special Simplex Implementations:

- The Revised Simplex Method. This is a systematic procedure for implementing the steps of the simplex method in a smaller array, thus saving storage space.
- The Decomposition Principle. This is a technique (sometimes called the Dantzig-Wolfe decomposition technique) for dealing with large-scale and/or specially structured linear programming problems.
- The Simplex Method for Bounded Variables. In some problems we have the bounded constraint $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$ instead of just the positivity constraint $\mathbf{x} \geq \mathbf{0}$. These bounded constraints could be treated by adding slack variables (i.e., $\mathbf{x} + \mathbf{s} = \mathbf{u}$ and $\mathbf{x} - \mathbf{e} = \mathbf{l}$), but this is inefficient for large problems since this increases the number of constraints and variables. The simplex method with bounded constraints handles these constraints more efficiently.
- Primal-Dual Methods. This method uses both the (primal) simplex method and the dual simplex method to move through the feasible region of the dual problem until primal feasibility is achieved. Some of the most recently developed linear programming algorithms use primal-dual methods.

Interior Point Methods This is a large and growing area of techniques that are often used instead of the Simplex Method. Unlike the Simplex Method, which starts on the boundary of the feasible region and traverses the boundary searching for the optimal point (due to convexity, the optimal point - if there is one - will lie on the boundary, in fact at a corner of the feasible region), interior point methods start in the interior of the feasible region and move towards the boundary. Clearly, moving *through* the feasible region could lead to finding the optimal point more quickly than traversing the

boundary. The key ingredients of interior point methods are; (1) determining which direction to move from a given point in the feasible region (often by looking at the gradient of the objective function at that point), (2) determining how far to move in that direction, and (3) testing whether the current point is sufficiently close to the optimal point ('stopping criteria'). Interior point methods include:

Karmarker's Projective Algorithm This is a polynomial-time algorithm developed in 1984 which is commonly used today.

Khachian's Ellipsoidal Algorithm Another polynomial-time algorithm (1979). However, this one has not proven to be any faster than the simplex algorithm in most applications and so it is not often used.

Game Theory:

Generally, game theory deals with the situation of 'players' competing for a limited amount of resources. Players have finitely many 'moves' available (the 'rules' of the game), and the 'pay-off' for any particular choice depends on the moves of the other players. One objective in the mathematical theory is to find 'optimal' strategies that assure a player of a certain pay-off regardless of what strategies the other players pursue. Hugely applicable in the field of economics. Much of the theory uses linear programming.

Nonlinear Programming:

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } x \in \Omega \end{aligned}$$

Here the feasible region is defined as $\Omega = \{g_1(x) \geq b_1, \cdots, g_k(x) \geq b_k,\ h_1(x) = c_1, \cdots, h_l(x) = c_l\}$. In a nonlinear programming problem, some (or all) of the functions $f(x), g_1(x), \cdots, g_k(x), h_1(x), \cdots, h_l(x)$ are nonlinear. An algorithm moves from one point x_n to the next x_{n+1} towards an optimal point.

Steepest Descent (or Ascent) If f is differentiable, then one looks for critical points of f. In one dimension, one can solve $f'(x) = 0$ by the bisection method or Newton's Method. In higher dimensions one can use the Method of Steepest Descent (for a min problem, or Ascent for a max problem); one moves in the direction of steepest decrease in f. This is given by the negative of the gradient: $-\nabla f(x)$. So $x_{n+1} = x_n - c\nabla f(x_n)$ for some $c > 0$ (remember that $x \in \mathbf{R}^n$, as is $\nabla f(x)$). What makes nonlinear programming much more difficult than linear programming is that there may be many *local* max or min; finding the *global* max or min could be nearly impossible for large problems. As in one dimension, we must check the boundary of Ω for local extrema of f in addition to critical points.

A modification of the method of steepest descent is the conjugate gradient method. Here one modifies the direction at each step so that in addition to being roughly parallel to the negative of the gradient of $f(x)$, it is 'noninterfering' (or 'conjugate') to the direction chosen in the previous step. Conjugate gradient methods are usually much faster than gradient methods.

Lagrange Multipliers In the case when there are only equality constraints (i.e., only the $h_i(x)$ constraints), then one can use the method of Lagrange multipliers. Here we look for critical points of the *Lagrangian function* $L(x_1, \cdots, x_n, \lambda_1, \cdots, \lambda_l) = f(x) - \sum_{i=0}^{l} \lambda_i(h_i(x) - c_i)$. By calculus, we try to solve the $n + l$ equations, $\partial L/\partial x_i = 0$, $\partial L/\partial \lambda_j = 0$ for $\bar{x}$ and $\bar{\lambda}$. Then $\bar{x}$ is a critical point of $f(x)$ subject to $h_i(x) = 0,\ i = 1, \cdots, l$, and $\nabla f(\bar{x}) = \sum_{i=1}^{l} \lambda_i \nabla h_i(\bar{x})$. However, as with critical points of any function, these $\bar{x}$ need not be local max or min (they could be saddle points). To determine whether they are or not (and which one of max or min they are), we need to look at the second derivatives (the *Hessian matrix*) of $f(x)$ (recall that the convexity of a function is related to its second derivatives). Incidentally, the Lagrangian function enables us to formulate a duality theory for nonlinear programming problems akin to that for linear programming problems.

Other Methods Trust region methods use a simpler approximation $\tilde{f}(x)$ to the objective function and a *trust region* about the point x for which this approximation is accurate. One solves this simpler problem on the trust region (i.e., find the next approximate solution x_{n+1}), and then repeats the procedure (i.e., find a trust region about x_{n+1}, etc). Penalty methods replace the 'hard' constraints $g_i(x) \leq 0,\ h_k(x) = 0$ with 'softer' constraints; the objective function is penalized if the point x gets close to the boundary of Ω (similar idea was used in the Big M method of linear programming). This way one is solving an *un*constrained problem (which in general are easier than constrained problems, although in this case we pay the price of making the objective function more complicated).

Special Nonlinear Problems If the objective function is a quadratic polynomial and the constraints are linear, the problem is called a quadratic programming problem. There are special algorithms that apply to this type of nonlinear programming problem.

Probabilistic Methods:

Many optimization problems in the real world are not defined precisely; some of the parameters are known only with some uncertainty (for example, arrival times of customers). These parameters become random variables rather than determined constants. Here one uses the tools of probability theory, in particular, stochastic processes (eg., Markov chains, random walks, Brownian motion, etc).

Another way probabilistic methods enter is in searching for solutions. The algorithms described above are *deterministic*; each step is precisely defined and predictable. In a probabilistic algorithm, some steps are choosen randomly (but with a purpose in mind!). For some problems, probabilistic methods are *much* faster than any deterministic algorithm. Many algorithms like this are referred to as *Monte Carlo Methods*.

Simulation:

With the advent of powerful computers another method became available to solve optimization problems. One can simulate outcomes of phenomena and vary the parameters to determine optimal outcomes (eg. traffic flow, where the parameters could be speed limits and traffic light patterns).

Dynamic Programming:

Here, a large optimization program is broken up into smaller problems, or sequential 'stages', which are solved with one (or more) of the techniques mentioned above. One begins at the last stage and works backwards to the beginning, solving each stage in an optimal way under the assumption that all preceding stages have been completed.

References

Bazaraa et. al., *Linear Programing and Network Flows.*

Bazaraa et. al., *Nonlinear Programing: Theory and Algorithms.*

R.E. Miller, *Optimization: Foundations and Applications.*

Nocedal, J. and Wright, S.J., *Numerical Optimization.*

A. Rapoport, *Two-Person Game Theory.*

S.M. Ross, *Introduction to Probability Models.*

W. Winston, *Operations Research.*