AMSC 607 / CMSC 764 Advanced Numerical Optimization

Fall 2008

UNIT 3: Constrained Optimization

PART 2: Feasible direction methods

Dianne P. O'Leary

Feasible Direction Methods

Reference: N&S Chapter 15

Idea:

- We have a collection of methods that work for unconstrained problems:
  - Newton's method
  - steepest descent
  - quasi-Newton
  - ...
- Can we modify them to work when we have constraints?

One advantage of these methods over Penalty methods (to be discussed later):
We only need to evaluate the function $f(\mathbf{x})$ for points in the feasible region.

This can be important if the function is undefined outside the feasible region or if we need to stop early.

Example: If $f(\mathbf{x})$ involves taking the $\log$ of one of the variables, then the function is undefined for zero and negative values of this variable. []

The plan:

- Handle linear equality constraints: generating feasible directions.
- Handle linear inequality constraints: active set strategies.
- Handle more general constraints:
  - sequential quadratic programming
  - reduced-gradient methods

Handling linear equality constraints

Problem:

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to

$$\mathbf{Ax} = \mathbf{b}$$

We'll assume that $\mathbf{A} \in \mathcal{R}^{m \times n}$.

We will assume rank($\mathbf{A}$) $= m < n$. This is a constraint qualification.

Suppose we have factored $\mathbf{A}$ as

$$\mathbf{A}^T = \mathbf{QR}$$

where $\mathbf{Q} \in \mathcal{R}^{n \times n}$ and $\mathbf{R} \in \mathcal{R}^{n \times m}$ is upper trapezoidal. Then the last $n - m$ columns of $\mathbf{Q}$ form a basis for the null space of $\mathbf{A}$.

How does this help us?

Once we have a basis for the nullspace of $\mathbf{A}$, we can express any feasible point as

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{Zv}$$

where $\mathbf{v} \in \mathcal{R}^{n-m}$ and $\hat{\mathbf{x}}$ solves the equations:

$$\mathbf{A}\hat{\mathbf{x}} = \mathbf{b}\,.$$

So now our minimization problem becomes

$$\min_{\mathbf{v}} f(\hat{\mathbf{x}} + \mathbf{Zv})$$

with no constraints at all!

So we can solve this using our old favorite methods.

Computational issues

The QR method guarantees that the condition number of the reduced Hessian matrix is no worse than the condition number of the Hessian matrix, since the columns of $\mathbf{Z}$ are orthonormal.

Handling linear inequality constraints

Problem:

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to

$$\mathbf{Ax} \geq \mathbf{b}$$

Suppose someone told you that at the optimal solution $\mathbf{x}^*$, the first k constraints held as equalities

$$\mathbf{A}_w\mathbf{x} = \mathbf{b}_w$$

(and $\mathbf{A}_w$ has full rank) but the other constraints were inactive:

$$\mathbf{A}_{\bar{w}}\mathbf{x} > \mathbf{b}_{\bar{w}}\,.$$

Then you could replace this problem by the equality constrained problem

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to

$$\mathbf{A}_w\mathbf{x} = \mathbf{b}_w.$$

Notation: We will let W be the index set for the set of active constraints and $\bar{W}$ be its complement. In the example we just did,

$$W = \{1, 2, \ldots, k\},\ \ \bar{W} = \{k+1, \ldots, m\}\,.$$

Our strategy

We can't expect someone to tell us what constraints are active at the solution point, but we can take a guess, reduce the function value, and repeat until we can't do any better.

Active set algorithms

The general scheme: Given an initial feasible point $\mathbf{x}^{(0)}$, set $k = 0$.
Until optimality,

1. Choose $W =$ the set of active constraint indices with $\mathbf{A}_w$ full rank.
2. Using $\mathbf{x}^{(k)}$ as an initial guess, solve

   $$\min_{\mathbf{x}} f(\mathbf{x})$$

   subject to

   $$\mathbf{A}_w\mathbf{x} = \mathbf{b}_w$$

   (or, at least step toward a solution to this problem) but don't allow the algorithm to walk outside the feasible set. Call your final point $\mathbf{x}^{(k+1)}$ and set $k = k + 1$.

Let's think about this in a bit more detail.

Finding a descent direction

We want to compute a descent direction for

$$\min_{\mathbf{v}} f(\mathbf{x} + \mathbf{Z}\mathbf{v})$$

where $\mathbf{x}$ is our current guess at the solution and $\mathbf{Z}$ is a basis for the nullspace of the matrix of the constraints that are active at $\mathbf{x}$.

- The Newton direction: The search direction for the v variables is

  $$(\mathbf{Z}^T\mathbf{H}(\mathbf{x})\mathbf{Z})\mathbf{p} = -\mathbf{Z}^T\mathbf{g}(\mathbf{x})$$

  (or, equivalently, $\mathbf{Zp}$ for the $\mathbf{x}$ variables)

- The steepest descent direction:

  $$\mathbf{p} = -\mathbf{Z}^T\mathbf{g}(\mathbf{x})$$

- Quasi-Newton direction:

  $$\mathbf{B}_w\mathbf{p} = -\mathbf{Z}^T\mathbf{g}(\mathbf{x})$$

  Complication: Every time we change the active set W, we have the issue of how to modify $\mathbf{B}_w$.

How far to walk?

Step in that direction until we satisfy the Wolfe (or G-A) linesearch conditions or until we encounter another constraint.

Unquiz: For Newton's method, we used an initial step length guess of 1. Now we need to reduce this length if another constraint is violated first. How can we compute how far can we step in direction v before encountering another constraint? []

Modifying W

Clearly, if we encounter another constraint on our step, we need to add it to the set of active constraints.

What if we don't encounter a constraint, and we succeed (eventually) in minimizing the function over the current set of active constraints W. Have we solved our problem?

Lagrange multipliers to the rescue

We need to check the optimality conditions for constrained optimization: is the gradient a nonnegative linear combination of the columns of $\mathbf{A}_w$?

So we are trying to solve the (possibly inconsistent) linear system

$$\mathbf{A}_w^T \boldsymbol{\lambda} = \mathbf{g}(\mathbf{x}) \,.$$

- If we have a QR factorization of $\mathbf{A}_w^T$, then the least squares solution to this problem is
$$\boldsymbol{\lambda} = {\mathbf{R}_1}^{-1} \mathbf{Q}^T \mathbf{g} \,,$$
where $\mathbf{R}_1$ is the square upper triangular matrix at the top of $\mathbf{R}$.
- We could use the partitioning method as an alternative to QR, partitioning $\mathbf{A}_w = [\mathbf{B}, \mathbf{N}]$, computing
$$\mathbf{Z} = \begin{bmatrix} -\mathbf{B}^{-1}\mathbf{N} \\ \mathbf{I} \end{bmatrix} ,$$
and letting
$$\boldsymbol{\lambda} = \begin{bmatrix} \mathbf{B}^{-T} & \mathbf{0} \end{bmatrix} \mathbf{g}(\mathbf{x}) \,.$$
Then
$$\begin{aligned} \mathbf{A}_w^T \boldsymbol{\lambda} &= \begin{bmatrix} \mathbf{B}^T \\ \mathbf{N}^T \end{bmatrix} \begin{bmatrix} \mathbf{B}^{-T} & \mathbf{0} \end{bmatrix} \mathbf{g}(\mathbf{x}) \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{N}^T\mathbf{B}^{-T} & \mathbf{0} \end{bmatrix} \mathbf{g}(\mathbf{x}) \end{aligned}$$
and $\boldsymbol{\lambda}$ is a solution to the first k equations.

Thus we have estimates for the current values of the Lagrange multipliers.

So now we know how to test optimality:

- If no constraints are active, we need to check that $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ and $\mathbf{H}(\mathbf{x})$ is positive definite.
- When constraints are active, we need to check that the Lagrange multipliers satisfy $\boldsymbol{\lambda} \geq \mathbf{0}$ and $\mathbf{A}_w^T \boldsymbol{\lambda} = \mathbf{g}$ and $\mathbf{Z}^T \mathbf{H}(\mathbf{x}) \mathbf{Z}$ is positive definite.

An important special case: Linear Programming

Example: linear programming

Consider this problem:

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x}$$

subject to

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}\,.$$

where $\mathbf{A} \in \mathcal{R}^{m \times n}$.

What does our algorithm look like in this special case?

We need to see how to

- solve the problem for the current set of active constraints.
- check optimality.
- choose a new set of active constraints.

Solving the problem for the current set of active constraints

Suppose we are currently at a point $\mathbf{x}$. Partition (and rearrange) $\mathbf{x}$ so that its nonzero components $\mathbf{x}_B$ come first, and rearrange the columns of $\mathbf{A}$ in the same way. Call the zero components $\mathbf{x}_N$, suppose it has $n - m$ components, and write the active constraints as

$$\begin{bmatrix} \mathbf{B} & \mathbf{N} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}$$

Then this is a system of n equations in n unknowns and has a unique solution as long as $\mathbf{B}$ is nonsingular.

Therefore, by solving this system, we determine the optimal solution for this set of active constraints without doing any minimization or line search.

Checking optimality

We need to compute the Lagrange multipliers

$$\begin{bmatrix} \mathbf{B}^T & \mathbf{0} \\ \mathbf{N}^T & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \end{bmatrix} = \mathbf{c}$$

so

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \end{bmatrix} &= \begin{bmatrix} \mathbf{B}^{-T} & \mathbf{0} \\ -\mathbf{N}^T\mathbf{B}^{-T} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{c}_B \\ \mathbf{c}_N \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{B}^{-T}\mathbf{c}_B \\ \mathbf{c}_N - \mathbf{N}^T\mathbf{B}^{-T}\mathbf{c}_B \end{bmatrix} \end{aligned}$$

To be optimal, we need $\boldsymbol{\lambda}_2 \geq \mathbf{0}$.

Jargon: The entries of $\boldsymbol{\lambda}_2$ are called the reduced costs.

($\boldsymbol{\lambda}_1$ is unrestricted in sign since its constraints are equality constraints.)

Choosing a new set of active constraints

Suppose that $\boldsymbol{\lambda}_2$ is not nonnegative.

Then we need to drop a constraint so that we can reduce the function value.

Which constraint can we drop? If we drop any constraint with a negative component in $\boldsymbol{\lambda}_2$, the function value will go down.

Customarily, we choose the constraint corresponding to the most negative multiplier.

This gives us a new set of active constraints, so we are ready for the next iteration.

A variant

Suppose we drop constraint j and let x_j be nonzero.

What is a feasible direction?

We need

$$\begin{bmatrix} \mathbf{B} & \mathbf{N} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{p} = \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_j \end{bmatrix}$$

or

$$\mathbf{p} = \begin{bmatrix} \mathbf{B}^{-1} & -\mathbf{B}^{-1}\mathbf{N} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_j \end{bmatrix} = \begin{bmatrix} -\mathbf{B}^{-1}\mathbf{N} \\ \mathbf{I} \end{bmatrix} \mathbf{e}_j \,.$$

Now that we have our direction, we can step along it until we hit a new constraint. Then we have n active constraints again, and we can repeat the process.

The Simplex Method for Linear Programming

The algorithm we just described is the simplex method for linear programming. The original variant is due to George Dantzig in the 1940s, building on work of many others, and there has been considerable effort since then. (See N&S p. 143, 180, 233.)

We've cleaned it up to present it in matrix terms, rather than the traditional tableau form.

Unquiz: Write up the algorithm.

Some interesting questions about the simplex method

- complexity.
- geometry: walk around edge.
- stability issues.
- history

Another important special case: Quadratic programming

In quadratic programming, we minimize a quadratic function subject to linear equality and inequality constraints.

- We encountered such problems in our discussion of trust regions.
- We'll see them in our SQP algorithm later in these notes.
- They are important in their own right.

It is a good exercise to work through what feasible direction algorithms look like in the special case of quadratic programming.

Handling nonlinear constraints

Two methods:

- sequential quadratic programming
- reduced gradient approach

Sequential quadratic programming

An almost feasible direction method.

A good reference: Boggs and Tolle, *Acta Numerica* 1995 p.1 or Nocedal & Wright Chapter 18.

SQP

An almost feasible point method.

Idea: For unconstrained optimization, we generate a search direction for

$$\min_{\mathbf{x}} f(\mathbf{x})$$

from a quadratic model $\mathbf{q}(\mathbf{x}+\mathbf{p})$ to f at $\mathbf{x}$.

This is Newton's method applied to $f(\mathbf{x})$.

So for constrained optimization, we'll try to generate a feasible direction for

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to

$$\mathbf{c}(\mathbf{x}) = \mathbf{0}$$

(linear or nonlinear constraints) from a quadratic model. We'll apply Newton's method to the Lagrangian

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x})\,.$$

There are two good ways to think about this.

Derivation 1: Quadratic model

We want a saddle point of the Lagrangian, minimizing with respect to $\mathbf{x}$ and maximizing with respect to $\boldsymbol{\lambda}$. Let's hold $\boldsymbol{\lambda}$ fixed. Then we have

$$L(\mathbf{x}^{(k)}+\mathbf{p}, \boldsymbol{\lambda}^{(k)}) \approx L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}) + (\bigtriangledown_x L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}))^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \bigtriangledown^2_{xx} L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)})\mathbf{p}$$

and

$$\mathbf{c}(\mathbf{x}^{(k)} + \mathbf{p}) \approx \mathbf{c}(\mathbf{x}^{(k)}) + \mathbf{A}(\mathbf{x}^{(k)})^T \mathbf{p}$$

So we choose to determine $\mathbf{p}$ by solving the quadratic programming problem

$$\min_{\mathbf{p}} (\bigtriangledown_x L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}))^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \bigtriangledown^2_{xx} L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)})\mathbf{p}$$

subject to

$$\mathbf{c}(\mathbf{x}^{(k)}) + \mathbf{A}(\mathbf{x}^{(k)})^T \mathbf{p} = \mathbf{0}\,.$$

Derivation 2: Newton's method

We want the gradient of the Lagrangian to be zero, so we apply Newton's method to

$$\bigtriangledown L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}\,.$$

This gives

$$\begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix}^{(k+1)} = \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix}^{(k)} + \alpha \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix}^{(k)}$$

where we define the directions $\mathbf{p}$ and $\mathbf{v}$ by solving the Newton equation

$$\bigtriangledown^2 L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}) \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix}^{(k)} = - \bigtriangledown L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}) \, .$$

What does this look like? $L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x})$, so

$$\bigtriangledown L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}) = \begin{bmatrix} \mathbf{g}(\mathbf{x}^{(k)}) - \mathbf{A}^T(\mathbf{x}^{(k)}) \boldsymbol{\lambda}^{(k)} \\ -\mathbf{c}(\mathbf{x}^{(k)}) \end{bmatrix}$$

$$\bigtriangledown^2 L(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}) = \begin{bmatrix} \mathbf{H}(\mathbf{x}^{(k)}) - \mathbf{G}(\mathbf{x}^{(k)}, \boldsymbol{\lambda}^{(k)}) & -\mathbf{A}^T(\mathbf{x}^{(k)}) \\ -\mathbf{A}(\mathbf{x}^{(k)}) & \mathbf{0} \end{bmatrix}$$

where $\mathbf{G}$ is the derivative with respect to $\mathbf{x}$ of $\mathbf{A}^T \boldsymbol{\lambda}$.

Fact: The Newton equation is the first order optimality condition for the quadratic programming problem in Derivation 1.

The SQP idea

Repeat until optimality:

- Generate a direction from solving the Newton equation.
- Take a step in that direction.

Advantages of SQP:

- Newton's method gives hope of quadratic convergence rate!
- Builds on well-understood technology: solving quadratic programming problems.
- One of the simpler algorithms for nonlinear constraints.

Disadvantages of SQP:

- It does not generate a sequence of feasible points, although it ultimately forces feasibility. (See, for example, the last column of Table 15.1 in N&S, p. 513.)
- Each iteration is very expensive and ordinarily we must compromise, making approximations:
  - Either neglect or approximate **G**.
  - Might use Quasi-Newton to approximate the (1,1) block of the matrix in the Newton equation.
  - Don't solve the QP; just insure sufficient progress. (But beware of trouble with feasibility if we stop early.)

Local expert

There are a lot of implementation issues that we have not discussed here.

Andre Tits (ECEE) is an expert on this method and has a very nice software package implementing it.

Reduced gradient methods

I don't think these methods have much of a future, but see N&S Section 15.6 if you are interested.

Final words

- Feasible direction methods and reduced gradient methods are losing popularity.
- SQP is still a good option.