Numerical Optimization Methods I

John MacLaren Walsh, Ph.D.

1 Reference

- **Nonlinear Programming, 2nd Ed.**, Dimitri P. Bertsekas. Athena Scientific, 1999.

Gradient Methods for Unconstrained Optimization

Steepest Descent

Suppose we are currently at a point $\mathbf{x}_k$ and would like to figure out where to move to next, staying locally within a small neighborhood of $\mathbf{x}_k$, say of size ϵ. Thus, we would like to select $\boldsymbol{\delta} \in \mathbf{B}_\epsilon(\mathbf{0})$, and then move to $\mathbf{x}_{k+1} = \mathbf{x} + \boldsymbol{\delta}$. The question is which direction $\boldsymbol{\delta}$ to select.

We could take a first order Taylor series expansion of the cost at $\mathbf{x}_k$ to get

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) + (\nabla f(\mathbf{x}))^T \boldsymbol{\delta} + o(\|\boldsymbol{\delta}\|) \tag{1}$$

Neglecting the $o(\epsilon)$ error term, the Cauchy-Schwartz inequality $|(\nabla f(\mathbf{x}_k))^T \boldsymbol{\delta}| \leq \|\nabla f(\mathbf{x}_k)\|_2 \|\boldsymbol{\delta}\|_2$ tells us that the first order change in the cost is made most negative when $\boldsymbol{\delta} = -\alpha \nabla f(\mathbf{x}_k)$ where $\alpha > 0$ is selected to scale $\boldsymbol{\delta}$ to lie within $\mathcal{B}_\epsilon(\mathbf{0})$ (i.e. $\alpha = \frac{\epsilon}{\|\nabla f(\mathbf{x}_k)\|_2}$).

The idea of repeatedly moving in small steps along the direction in which the cost decreases the most to first order is that of a *steepest descent* algorithm, and can be written as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) \tag{2}$$

Newton's Method

We can develop Newton's method in a similar manner to that of steepest descent by starting with a second order Taylor series instead of a first order one.

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) + (\nabla f(\mathbf{x}_k))^T (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x}_{k+1} - \mathbf{x}_k)^T \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) + o(\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2) \tag{3}$$

Defining $\boldsymbol{\delta} = \mathbf{x}_{k+1} - \mathbf{x}_k$ we have

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) + (\nabla f(\mathbf{x}_k))^T \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T \nabla^2 f(\mathbf{x}_k)\boldsymbol{\delta} + o(\|\boldsymbol{\delta}\|^2) \tag{4}$$

Neglecting the higher order terms, and provided that $\nabla^2 f(\mathbf{x}_k) > \mathbf{0}$ (i.e. that the Hessian matrix here is positive definite) we find that the $\boldsymbol{\delta}$ which provides the largest decrease by taking the gradient with respect to $\boldsymbol{\delta}$ to find that it solves the equation

$$(\nabla^2 f(\mathbf{x}_k))\boldsymbol{\delta} + \nabla f(\mathbf{x}_k) = \mathbf{0} \tag{5}$$

This suggests the iterative algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) \tag{6}$$

General Gradient Optimization Algorithm Form

Interpolating between these two simplest case algorithms (steepest descent and Newton), we get a "general" algorithm form for Gradient optimization algorithms.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{D}_k \boldsymbol{\delta}_k \tag{7}$$

Here the sequence of scalars α_k are referred to as the "step size", $\mathbf{D}_k$ is a positive definite matrix ($(\nabla^2 f(\mathbf{x}_k))^{-1}$ for a general Newton's method), and $\boldsymbol{\delta}_k$ is a suitably chosen direction (e.g. $\nabla f(\mathbf{x}_k)$).

Many algorithms for unconstrained optimizations take variations on the Newton/steepest descent theme by minor modifications within this broad form. For instance, sometimes instead of inverting the second order derivative matrix a diagonal or block diagonal approximation is used to create $\mathbf{D}_k$, etc. Alternatively the gradient and or Hessian may be numerically approximated, or only a noisy estimate of the gradient may be available, etc.

Selecting the Step Size

Given that we have introduced the basic steepest descent and Newton forms, it is now of interest how to determine the only remaining parameter when applying these methods, the step size α_k. Step size selection can be subtle and problem dependent. Three methods that are especially ofa interest

- **Constant** $\alpha_k = c$
- **Decreasing** $\alpha_k \to 0$ with $\sum_k \alpha_k \to \infty$
- **Armijo** $\alpha_k = \beta^{m_k} s$, $\beta, s \in (0,1)$ with m_k selected so that

$$f(\mathbf{x}_k) - f(\mathbf{x}_k + \beta^{m_k} s \boldsymbol{\delta}_k) \geq -\sigma \beta^{m_k} s (\nabla f(\mathbf{x}_k))^T \boldsymbol{\delta}_k \tag{8}$$

The crudest of all the step size rules, selecting a constant, is typically used when the optimization algorithm is being run online with a sequence of noisy cost gradients yielding an actual minimum that is slowly changing and must be tracked, or alternatively can be employed for the sake of its simplicity. In these cases, it must be selected carefully, as it must small enough to ensure that the algorithm does not diverge, but large enough to allow for near convergence within a reasonable number of iterations and for accurate tracking of the slowly varying minimum.

Decreasing step sizes are especially useful when the gradient being employed is only a noisy estimate of the true gradient of a cost with a stationary minimum. In this case, it is desirable for the step size to decrease so that the amount of rattle caused by the additive disturbance to the gradient will be decreased as the optimum is approached. On the other hand, care must be taken in order to avoid stopping, due to a very small step size, before a minima is reached. In this regard, sequences which not only have the property $\alpha_k \to 0$, but also the property $\sum_k \alpha_k \to \infty$, so as to ensure no pre-mature stopping, are useful.

The Armijo rule is an especially useful step size rule for fast convergence when the cost can be directly computed (as is the case in many applications of nonlinear programming techniques).

Convergence

Provided a search direction $\mathbf{D}_k \boldsymbol{\delta}_k$ which can sufficiently decrease the cost and an appropriate initialization, the Armijo rule can guarantee convergence to a local minimum with a local (in the close vicinity of the minimum) convergence that is superlinear (i.e. with error that is $o(\frac{1}{k})$ for each the steepest descent or the Newton method. The relevant analysis, and extension of these ideas to other step size rules and algorithms can be found circa page 40 and circa page 80 in Bertsekas.

Handling Constraints

Given the availability of efficient unconstrained local gradient based optimization algorithms, it is of interest to determine how these can be extended to handle constrained optimization scenarios.

Inequality Constraints and the Barrier Method

Let's begin by considering problems with inequality constraints $g_j(\mathbf{x}) \leq 0,\ j \in \{1, \ldots, J\}$. If there is a point in the interior of these constraints, so that $g_j(\mathbf{x}) < 0,\ j \in \{1, \ldots, J\}$, then the barrier method provides an effective technique for dealing with them. The idea is to add a term to the cost which diverges to ∞ as $\mathbf{x}$ approaches a point for which one of the constraints $g_j(\mathbf{x}) = 0$. Two possibilities for such terms include

$$-\sum_{j=1}^{J} \frac{1}{g_j(\mathbf{x})} \quad \text{and} \quad -\sum_{j=1}^{J} \log(-g_j(\mathbf{x})) \tag{9}$$

These terms are multiplied with a constant ϵ_ℓ and added to the cost function $f(\mathbf{x})$ to get an aggregate function, i.e.

$$J_\ell(\mathbf{x}) = f(\mathbf{x}) - \epsilon_\ell \sum_{j=1}^{J} \log(-g_j(\mathbf{x})) \tag{10}$$

This cost is minimized with an unconstrained optimization algorithm (such as steepest descent or Newton's method) for progressive ϵ_ℓ (the initialization for the next ℓ is the convergent point of the previous ℓ) with $\epsilon_\ell \to 0$. Given reasonable regularity assumptions on the inequality constraints, this technique ensures convergence of the solution to the optimization problem to a constrained local optimum.

Equality Constraints and the Penalty Method

One technique, known as the penalty method, for handling equality constraints in numerical optimization methods bears resemblance to the barrier method for inequality constraints. The basic idea is to add to the cost a term penalizing deviation from the constraint set. Namely, suppose that the equality constraints are $h_i(\mathbf{x}) = 0,\ i \in \{1, \ldots, I\}$, and the cost function is $f(\mathbf{x})$. One forms a sequence of augmented costs such as

$$J_\ell(\mathbf{x}) = f(\mathbf{x}) + \gamma_\ell \sum_{i=1}^{I} (h_i(\mathbf{x}))^2 \tag{11}$$

with a sequence of constants $\gamma_\ell \to \infty$. For each γ_ℓ a gradient optimization algorithm is run initialized at the result of locally minimizing the augmented cost from the previous γ_ℓ. Under a set of reasonable regularity conditions for the constraint functions h_i the sequence of solutions will converge to a local optimum of the constrained problem as $\gamma_\ell \to \infty$.