Numerical Optimization

Emo Todorov

Applied Mathematics and Computer Science & Engineering

University of Washington

Spring 2010

Gradient descent

The directional derivative of $f(\mathbf{x})$ at $\mathbf{x}_0$ in direction $\mathbf{v}$ is

$$D_{\mathbf{v}}[f](\mathbf{x}_0) = \left.\frac{df(\mathbf{x}_0 + \varepsilon\mathbf{v})}{d\varepsilon}\right|_{\varepsilon=0}$$

Let $\mathbf{x}(\varepsilon) = \mathbf{x}_0 + \varepsilon\mathbf{v}$. Then $f(\mathbf{x}_0 + \varepsilon\mathbf{v}) = f(\mathbf{x}(\varepsilon))$ and the chain rule yields

$$D_{\mathbf{v}}[f](\mathbf{x}_0) = \left.\frac{\partial\mathbf{x}(\varepsilon)^{\mathsf{T}}}{\partial\varepsilon}\right|_{\varepsilon=0} \left.\frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{v}^{\mathsf{T}} \left.\frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{v}^{\mathsf{T}}\mathbf{g}(\mathbf{x}_0)$$

where $\mathbf{g}$ denotes the gradient of f.

Gradient descent

The directional derivative of $f(\mathbf{x})$ at $\mathbf{x}_0$ in direction $\mathbf{v}$ is

$$D_{\mathbf{v}}[f](\mathbf{x}_0) = \left.\frac{df(\mathbf{x}_0 + \varepsilon\mathbf{v})}{d\varepsilon}\right|_{\varepsilon=0}$$

Let $\mathbf{x}(\varepsilon) = \mathbf{x}_0 + \varepsilon\mathbf{v}$. Then $f(\mathbf{x}_0 + \varepsilon\mathbf{v}) = f(\mathbf{x}(\varepsilon))$ and the chain rule yields

$$D_{\mathbf{v}}[f](\mathbf{x}_0) = \left.\frac{\partial\mathbf{x}(\varepsilon)^{\mathsf{T}}}{\partial\varepsilon}\right|_{\varepsilon=0} \left.\frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{v}^{\mathsf{T}} \left.\frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{v}^{\mathsf{T}}\mathbf{g}(\mathbf{x}_0)$$

where $\mathbf{g}$ denotes the gradient of f.

Theorem (steepest ascent direction)

The maximum of $D_{\mathbf{v}}[f](\mathbf{x}_0)$ s.t. $\|\mathbf{v}\| = 1$ is achieved when $\mathbf{v}$ is parallel to $\mathbf{g}(\mathbf{x}_0)$.

Gradient descent

The directional derivative of $f(\mathbf{x})$ at $\mathbf{x}_0$ in direction $\mathbf{v}$ is

$$D_{\mathbf{v}}[f](\mathbf{x}_0) = \left.\frac{df(\mathbf{x}_0 + \varepsilon\mathbf{v})}{d\varepsilon}\right|_{\varepsilon=0}$$

Let $\mathbf{x}(\varepsilon) = \mathbf{x}_0 + \varepsilon\mathbf{v}$. Then $f(\mathbf{x}_0 + \varepsilon\mathbf{v}) = f(\mathbf{x}(\varepsilon))$ and the chain rule yields

$$D_{\mathbf{v}}[f](\mathbf{x}_0) = \left.\frac{\partial \mathbf{x}(\varepsilon)^{\mathsf{T}}}{\partial \varepsilon}\right|_{\varepsilon=0} \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{v}^{\mathsf{T}} \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{v}^{\mathsf{T}}\mathbf{g}(\mathbf{x}_0)$$

where $\mathbf{g}$ denotes the gradient of f.

Theorem (steepest ascent direction)

The maximum of $D_{\mathbf{v}}[f](\mathbf{x}_0)$ s.t. $\|\mathbf{v}\| = 1$ is achieved when $\mathbf{v}$ is parallel to $\mathbf{g}(\mathbf{x}_0)$.

Algorithm (gradient descent)

Set $\mathbf{x}_{k+1} = \mathbf{x}_k - \beta_k \mathbf{g}(\mathbf{x}_k)$ where β_k is the step size. The optimal step size is

$$\beta_k^* = \arg\min_{\beta_k} f(\mathbf{x}_k - \beta_k \mathbf{g}(\mathbf{x}_k))$$

Line search

Most optimization methods involve an inner loop which seeks to minimize (or sufficiently reduce) the objective function constrained to a line: $f(\mathbf{x} + \varepsilon \mathbf{v})$, where $\mathbf{v}$ is such that a reduction in f is always possible for sufficiently small ε, unless f is already at a local minimum. In gradient descent $\mathbf{v} = -\mathbf{g}(\mathbf{x})$; other choices are possible (see below) as long as $\mathbf{v}^{\mathsf{T}} \mathbf{g}(\mathbf{x}) \leq 0$.

Line search

Most optimization methods involve an inner loop which seeks to minimize (or sufficiently reduce) the objective function constrained to a line: $f(\mathbf{x} + \varepsilon \mathbf{v})$, where $\mathbf{v}$ is such that a reduction in f is always possible for sufficiently small ε, unless f is already at a local minimum. In gradient descent $\mathbf{v} = -\mathbf{g}(\mathbf{x})$; other choices are possible (see below) as long as $\mathbf{v}^\mathsf{T} \mathbf{g}(\mathbf{x}) \leq 0$.

This is called *linesearch*, and can be done in different ways:

1. Backtracking: try some ε, if $f(\mathbf{x} + \varepsilon \mathbf{v}) > f(\mathbf{x})$ reduce ε and try again.
2. Bisection: attempt to minimize $f(\mathbf{x} + \varepsilon \mathbf{v})$ w.r.t. ε using a bisection method.
3. Polysearch: attempt to minimize $f(\mathbf{x} + \varepsilon \mathbf{v})$ by fitting quadratic or cubic polynomials in ε, finding the minimum analytically, and iterating.

Exact minimization w.r.t. ε is often a waste of time because for $\varepsilon \neq 0$ the current search direction may no longer be a descent direction.

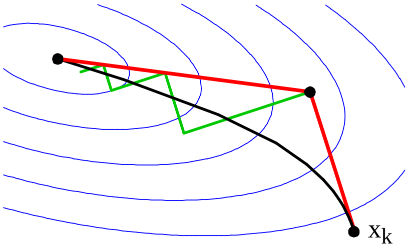
Sufficient reduction in f is defined relative to the local model (linear or quadratic). This is known as the Armijo-Goldstein condition; the Wolfe condition (which also involves the gradient) is more complicated.

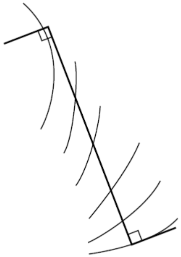
Chattering

If $\mathbf{x}_{k+1}$ is a (local) minimum of f in the search direction $\mathbf{v}_k = -\mathbf{g}(\mathbf{x}_k)$, then $D_{\mathbf{v}_k}[f](\mathbf{x}_{k+1}) = 0 = \mathbf{v}_k^{\mathsf{T}} \mathbf{g}(\mathbf{x}_{k+1})$, and so if we use $\mathbf{v}_{k+1} = -\mathbf{g}(\mathbf{x}_{k+1})$ as the next search direciton, we have $\mathbf{v}_{k+1}$ orthogonal to $\mathbf{v}_k$. Thus gradient descent with exact line search (i.e. steepest descent) makes a 90 deg turn at each iteration, which causes chattering when the function has a long oblique valey.

Chattering

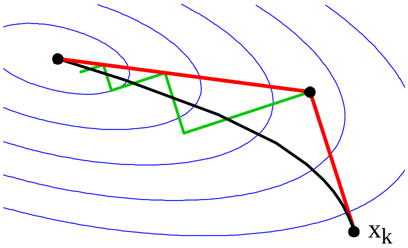
If $\mathbf{x}_{k+1}$ is a (local) minimum of f in the search direction $\mathbf{v}_k = -\mathbf{g}(\mathbf{x}_k)$, then $D_{\mathbf{v}_k}[f](\mathbf{x}_{k+1}) = 0 = \mathbf{v}_k^{\mathsf{T}}\mathbf{g}(\mathbf{x}_{k+1})$, and so if we use $\mathbf{v}_{k+1} = -\mathbf{g}(\mathbf{x}_{k+1})$ as the next search direciton, we have $\mathbf{v}_{k+1}$ orthogonal to $\mathbf{v}_k$. Thus gradient descent with exact line search (i.e. steepest descent) makes a 90 deg turn at each iteration, which causes chattering when the function has a long oblique valey.

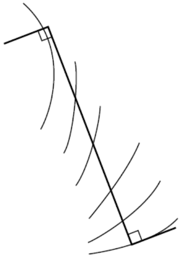




Chattering

If $\mathbf{x}_{k+1}$ is a (local) minimum of f in the search direction $\mathbf{v}_k = -\mathbf{g}(\mathbf{x}_k)$, then $D_{\mathbf{v}_k}[f](\mathbf{x}_{k+1}) = 0 = \mathbf{v}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_{k+1})$, and so if we use $\mathbf{v}_{k+1} = -\mathbf{g}(\mathbf{x}_{k+1})$ as the next search direciton, we have $\mathbf{v}_{k+1}$ orthogonal to $\mathbf{v}_k$. Thus gradient descent with exact line search (i.e. steepest descent) makes a 90 deg turn at each iteration, which causes chattering when the function has a long oblique valey.





Key to developing more efficient methods is to anticipate how the gradient will rotate as we move along the current search direction.

Newton's method

Theorem

If all you have is a hammer, then everything looks like a nail.

Corollary

If all you can optimize is a quadratic, then every function looks like a quadratic.

Newton's method

Theorem

If all you have is a hammer, then everything looks like a nail.

Corollary

If all you can optimize is a quadratic, then every function looks like a quadratic.

Taylor-expand $f(\mathbf{x})$ around the current solution $\mathbf{x}_k$ up to 2nd order:

$$f(\mathbf{x}_k + \boldsymbol{\varepsilon}) = f(\mathbf{x}_k) + \boldsymbol{\varepsilon}^\mathsf{T}\mathbf{g}(\mathbf{x}_k) + \frac{1}{2}\boldsymbol{\varepsilon}^\mathsf{T} H(\mathbf{x}_k)\,\boldsymbol{\varepsilon} + o\left(\boldsymbol{\varepsilon}^3\right)$$

where $\mathbf{g}(\mathbf{x}_k)$ and $H(\mathbf{x}_k)$ are the gradient and Hessian of f at $\mathbf{x}_k$:

$$\mathbf{g}(\mathbf{x}_k) \triangleq \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k} \qquad H(\mathbf{x}_k) \triangleq \left.\frac{\partial^2 f}{\partial \mathbf{x}\partial \mathbf{x}^\mathsf{T}}\right|_{\mathbf{x}=\mathbf{x}_k}$$

Assuming H is (symmetric) positive definite, the next solution is

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \arg\min_{\boldsymbol{\varepsilon}} \left\{\boldsymbol{\varepsilon}^\mathsf{T}\mathbf{g} + \frac{1}{2}\boldsymbol{\varepsilon}^\mathsf{T} H\boldsymbol{\varepsilon}\right\} = \mathbf{x}_k - H^{-1}\mathbf{g}$$

Stabilizing Newton's method

For *convex* functions the Hessian H is always s.p.d, so the above method converges (usually quickly) to the global minimum. In reality however most functions we want to optimize are non-convex, which causes two problems:

1. H may be singular, which means that $\mathbf{x}_{k+1} = \mathbf{x}_k - H^{-1}\mathbf{g}$ will take us all the way to infinity.
2. H may have negative eigenvalues, which measn that (even if $\mathbf{x}_{k+1}$ is finite) we end up finding saddle points – minimum in some directions, maximum in other directions.

Stabilizing Newton's method

For *convex* functions the Hessian H is always s.p.d, so the above method converges (usually quickly) to the global minimum. In reality however most functions we want to optimize are non-convex, which causes two problems:

1. H may be singular, which means that $\mathbf{x}_{k+1} = \mathbf{x}_k - H^{-1}\mathbf{g}$ will take us all the way to infinity.
2. H may have negative eigenvalues, which measn that (even if $\mathbf{x}_{k+1}$ is finite) we end up finding saddle points – minimum in some directions, maximum in other directions.

These problems can be avoided in two general ways:

1. Trust region: minimize $\boldsymbol{\varepsilon}^\mathsf{T}\mathbf{g} + \frac{1}{2}\boldsymbol{\varepsilon}^\mathsf{T}H\boldsymbol{\varepsilon}$ s.t. $\|\boldsymbol{\varepsilon}\| \leq r$, where r is adapted over iterations. The minimization is usually done approximately.
2. Convexification/linearsearch: replace H with $H + \lambda I$, and/or use backtracking linesearch starting at the Newton point. When λ is large, $\mathbf{x}_k - (H + \lambda I)^{-1}\mathbf{g} \approx \mathbf{x}_k - \lambda^{-1}\mathbf{g}$, which is gradient descent with step λ^{-1}. The Levenberg-Marquardt method adapts λ over iterations.

Relation to linear solvers

The quadratic function

$$f\left(\mathbf{x}_k + \boldsymbol{\varepsilon}\right) = f\left(\mathbf{x}_k\right) + \boldsymbol{\varepsilon}^{\mathsf{T}}\mathbf{g}\left(\mathbf{x}_k\right) + \frac{1}{2}\boldsymbol{\varepsilon}^{\mathsf{T}}H\left(\mathbf{x}_k\right)\boldsymbol{\varepsilon}$$

is minimized when the gradient w.r.t $\boldsymbol{\varepsilon}$ vanishes, i.e. when

$$H\boldsymbol{\varepsilon} = -\mathbf{g}$$

When H is s.p.d, one can use the conjugate-gradient method for solving linear equations to do numerical optimization.

The set of vectors $\{\mathbf{v}_k\}_{k=1\cdots n}$ are conjugate if they satisfy $\mathbf{v}_i^{\mathsf{T}}H\mathbf{v}_j = 0$ for $i \neq j$. These are good search directions because they yield exact minimization of an n-dimensional quadratic in n iterations (using exact linesearch). Such a set can be constructed using Lanczos iteration:

$$s_{k+1}\mathbf{v}_{k+1} = \left(H - \alpha_k I\right)\mathbf{v}_k - s_k\mathbf{v}_{k-1}$$

where s_{k+1} is such that $\|\mathbf{v}_{k+1}\| = 1$, and $\alpha_k = \mathbf{v}_k^{\mathsf{T}}H\mathbf{v}_k$. Note that access to H is not required; all we need to be able to compute is $H\mathbf{v}$.

Non-linear least squares

Many optimization problems are in the form

$$f(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2$$

where $\mathbf{r}(\mathbf{x})$ is a vector of "residuals". Define the Jacobian of the residuals:

$$J(\mathbf{x}) = \frac{\partial \mathbf{r}(\mathbf{x})}{\partial \mathbf{x}}$$

Then the gradient and Hessian of f are

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &= J(\mathbf{x})^\mathsf{T}\mathbf{r}(\mathbf{x}) \\ H(\mathbf{x}) &= J(\mathbf{x})^\mathsf{T} J(\mathbf{x}) + \frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} \times \mathbf{r}(\mathbf{x}) \end{aligned}$$

We can omit the last term and obtain the Gauss-Netwon approximation:

$$H(\mathbf{x}) \approx J(\mathbf{x})^\mathsf{T} J(\mathbf{x})$$

Then Newton's method (with stabilization) becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(J_k^\mathsf{T} J_k + \lambda_k I\right)^{-1} J_k^\mathsf{T}\mathbf{r}_k$$