# NUMERICAL OPTIMIZATION IN THE CONTEXT OF CLASSIFICATION PROBLEMS

MARIA CAMERON

Contents



## 1. An overview of algorithms for unconstrained optimization

Read Chapter 2 of Nocedal and Wright, "Numerical Optimization" [1]. See Fig. 1.

## 2. Line Search Methods

Read Chapter 3 of Nocedal and Wright "Numerical Optimization" [1]. Keywords:

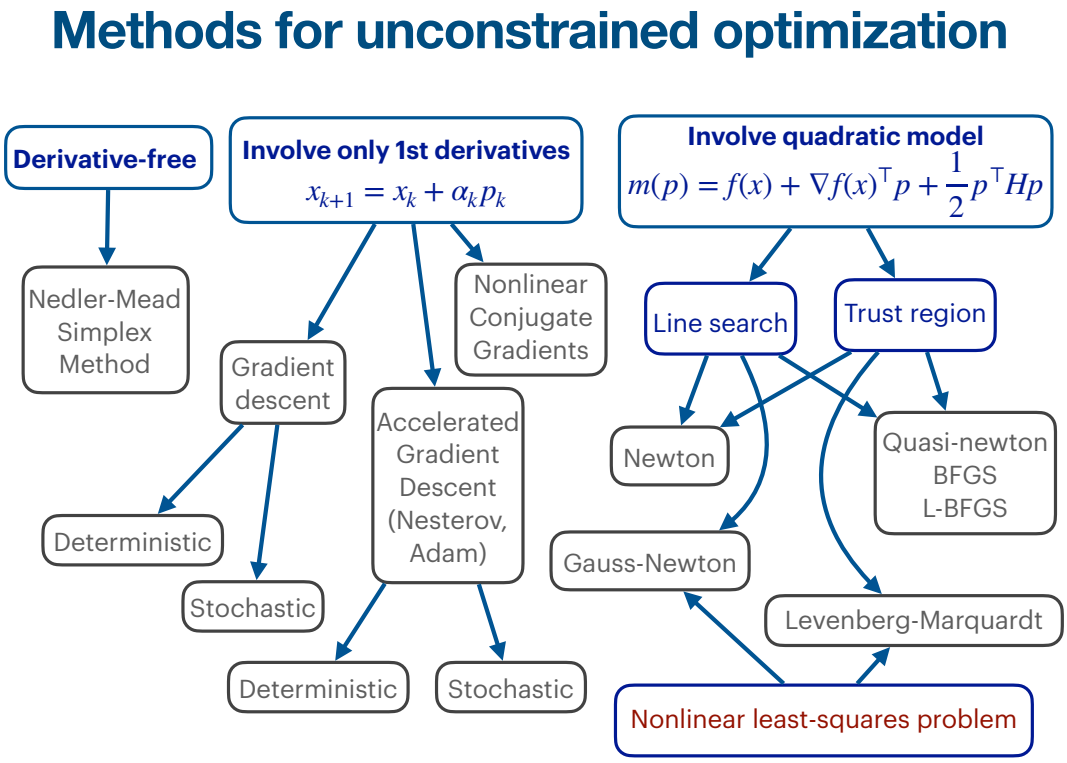


FIGURE 1. Classification of algorithms for unconstrained optimization.

- the Wolfe conditions,
- steepest descent,
- Newton's method,
- BFGS,
- convergence.

## 3. Trust-Region Methods

Read Chapter 4 of Nocedal and Wright "Numerical Optimization" [1].

## 4. BFGS and L-BFGS

4.1. **BFGS.** BFGS (Broyden-Fletcher-Goldfarb-Shanno) is, perhaps, the most successful quasi-newton method [1] (Section 2.2). At each step of optimizing a function $f(\mathbf{x})$, any quasi-newton method updates a quadratic model for it

$$m(\mathbf{p}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top B_k \mathbf{p}, \tag{1}$$

where $B_k$ is a matrix approximating the Hessian of $f$. The step direction $\mathbf{p}$ is the minimizer of (1):

$$\mathbf{p}_k = -B_k^{-1}\nabla f(\mathbf{x}_k).$$

The matrix $B_k$ is constructed as follows. The initial matrix is often set to identity: $B_0 = I$. Then, at each step, is it updated to match the action of the actual Hessian of $f$ on the actual step. For brevity, we will denote $f_k \equiv f(\mathbf{x}_k)$, $\nabla f_k \equiv \nabla f(\mathbf{x}_k)$, and $\nabla\nabla f(\mathbf{x}_k) \equiv \nabla\nabla f_k$. Taylor expansion at $\mathbf{x}_k$ yields the following identity:

$$\nabla f_{k+1} = \nabla f_k + \nabla\nabla f_k(\mathbf{x}_{k+1} - \mathbf{x}_k) + o(\|\mathbf{x}_{k+1} - \mathbf{x}_k\|). \tag{2}$$

Hence

$$\nabla f_{k+1} - \nabla f_k \approx \nabla\nabla f_k(\mathbf{x}_{k+1} - \mathbf{x}_k). \tag{3}$$

We introduce notation

$$\mathbf{s}_k := \mathbf{x}_{k+1} - \mathbf{x}_k \quad \text{and} \quad \mathbf{y}_k := \nabla f_{k+1} - \nabla f_k$$

and define the update requirement for the matrix $B$ motivated by (3):

$$B_{k+1}\mathbf{s}_k = \mathbf{y}_k. \tag{4}$$

Note that $B$ is $d \times d$ while (4) gives only $d$ equations. Therefore, (4) is an underdetermined system that has a $d$-dimensional solution space. The *BFGS update formula* for $B_k$ adds a matrix of rank 2 to it designed so that $B_k$ remains symmetric positive definite provided that $B_0$ is symmetric positive definite and $\mathbf{s}_k^\top \mathbf{y}_k > 0$ for all $k$.

**Exercise** Prove that all matrices $B_k$ generated by BFGS

$$B_{k+1} = B_k - \frac{B_k\mathbf{s}_k\mathbf{s}_k^\top B_k}{\mathbf{s}_k^\top B_k \mathbf{s}_k} + \frac{\mathbf{y}_k\mathbf{y}_k^\top}{\mathbf{y}_k^\top \mathbf{s}_k} \tag{5}$$

are symmetric positive definite provided that such is $B_0$ and $\mathbf{s}_k^\top \mathbf{y}_k > 0$ for all $k \in \mathbb{N}$.

A convenient feature of BFGS is that the inverse matrices $H_k \equiv B_k^{-1}$ can be generated automatically instead of $B_k$:

$$H_{k+1} = V_k^\top H_k V_k + \rho_k \mathbf{s}_k\mathbf{s}_k^\top, \quad \text{where} \quad \rho_k = \frac{1}{\mathbf{y}_k^\top \mathbf{s}_k}, \quad V_k = I - \rho_k \mathbf{y}_k\mathbf{s}_k^\top. \tag{6}$$

**Exercise** Prove that the matrices $H_k$ given by (6) are inverses of $B_k$ given by (5).

Fig. 2 displays iterates produced by three line search methods: Newton's (blue), BFGS (dark green), and gradient descent (dark red) applied to the Rosenbrock function

$$f(\mathbf{x}) = (1 - x_1)^2 + 5(x_2 - x_1^2)^2 \quad \text{with initial guess} \quad [-1.3, 1.5]^\top. \tag{7}$$

This function has a unique local minimum at $[1; 1]^\top$ but with the given parameter values it is not convex. For Newton's method, the Hessian is modified in the case if it is not positive definite – see the code below. The stopping criterion is $\|\nabla f\| < 10^{-10}$. Newton's method converges in 10 iterations, BFGS – in 18, while gradient descent takes 270 iterations most of which are in a small neighborhood of the solution. The main lesson for us is that *BFGS converges almost as fast as Newton as iterates get to a neighborhood of the local minimum in which the objective function is well-approximated by a convex quadratic.*

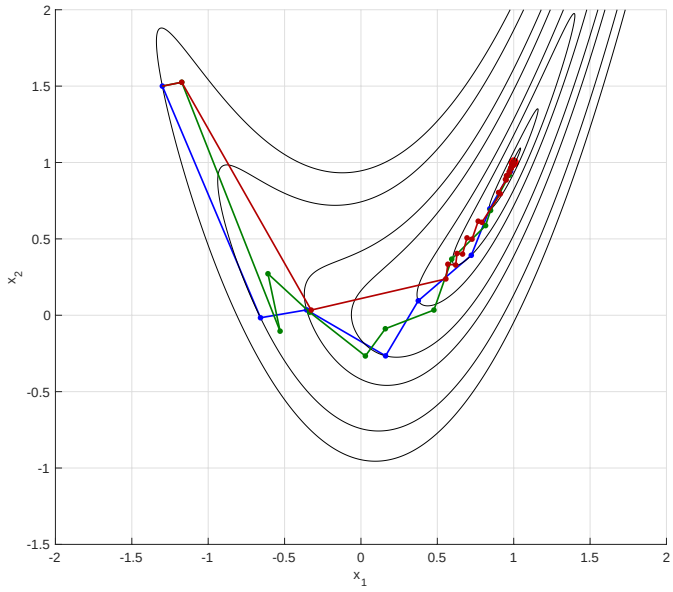


FIGURE 2. Iterates of Newton's (blue), BFGS (dark green), and gradient descent (dark red) applied to (7).

4.2. **L-BFGS.** L-BFGS stands for *limited memory BFGS* [1] (Section 7.2). It is often the method of choice for large-scale problems where the Hessian cannot be computed at a reasonable cost or is not sparse. L-BFGS (as other limited-memory quasi-Newton methods) stores a small fixed number of vectors (e.g., $m = 5$) that represent an approximation to the Hessian implicitly, i.e., it stores the pairs $(\mathbf{s}_i, \mathbf{y}_i)$ for $i = k-m, \ldots, k-1$. At each iteration $k$, an initial approximation $H_k^0$ to the inverse Hessian is chosen. One such approximation that has proven effective in practice is (see [1] Section 6.1)

$$H_k^0 = \frac{\mathbf{s}_{k-1}^\top \mathbf{y}_{k-1}}{\mathbf{y}_{k-1}^\top \mathbf{y}_{k-1}}. \tag{8}$$

Then this approximation is updated by

$$H_k^{j+1} = V_{k-m+j}^\top H_k^j V_{k-m+j} + \rho_{k-m+j} \mathbf{s}_{k-m+j} \mathbf{s}_{k-m+j}^\top, \quad j = 0, \ldots, m-1, \tag{9}$$

where $\rho_i$ and $V_i$ are defined in (6). Equation (9) suggests that the matrix-vector multiplication defining the direction of the step $\mathbf{p}_k = -H_k^m \nabla f_k$ can be performed in two for-loops implemented in the Matlab code below.

```
function p = finddirection(g,s,y,rho)
% input: g = gradient dim-by-1
% s = matrix dim-by-m, s(:,i) = x_{k-i+1}-x_{k-i}
```

```
% y = matrix dim-by-m, y(:,i) = g_{k-i+1}-g_{k-i}
% rho is 1-by-m, rho(i) = 1/(s(:,i)'*y(:,i))
m = size(s,2);
a = zeros(m,1);
for i = 1 : m
    a(i) = rho(i)*s(:,i)'*g;
    g = g - a(i)*y(:,i);
end
gam = s(:,1)'*y(:,1)/(y(:,1)'*y(:,1)); % H0 = gam*eye(dim)
g = g*gam;
for i = m :-1 : 1
    aux = rho(i)*y(:,i)'*g;
    g = g + (a(i) - aux)*s(:,i);
end
p = -g;
end
```

L-BFGS keeps in memory pairs of vectors $(\mathbf{s}_k, \mathbf{y}_k)$ from the most recent $m$ steps and replaces the least recent pair with the new pair at each step. The Matlab program below encodes L-BFGS and testing it on the Rosenbrock function (7). The convergence is achieved in 20 iterations which is comparable with Newton's and close to BFGS (10 and 18, respectively) and which is much fewer than gradient descent (270). The majority of these iterates are done in a small neighborhood of the minimizer. Fig. 3 is generated by this routine. The norm of the gradient versus iteration number for all four methods are plotted in Fig. 4.

```
function LBFGS()
%%  the Rosenbrock function and parameters
a = 5;
func = @(x,y)(1-x).^2 + a*(y - x.^2).^2;  % Rosenbrock's function
gfun = @(x)[-2*(1-x(1))-4*a*(x(2)-x(1)^2)*x(1);2*a*(x(2)-x(1)^2)]; % gradient of f
Hfun = @(x)[2 + 12*a*x(1)^2 - 4*a*x(2), -4*a*x(1); -4*a*x(1), 2*a]; % Hessian of f
gam = 0.9; % line search step factor
jmax = ceil(log(1e-14)/log(gam)); % max # of iterations in line search
eta = 0.5; % backtracking stopping criterion factor
tol = 1e-10;
m = 5; % the number of steps to keep in memory
%%
close all
figure;
hold on; grid;
x0 = [-1.3;1.5];  %initial guess
xstar = [1;1]; % the global minimizer
[xx,yy]=meshgrid(linspace(-2,2,1000),linspace(-1.5,2,1000));
ff = func(xx,yy);
```

```
plot(xstar(1),xstar(2),'r.','Markersize',40);
daspect([1,1,1])
col = [0.4,0.2,0];
%
s = zeros(2,m);
y = zeros(2,m);
rho = zeros(1,m);
%
x = x0;
g = gfun(x);
plot(x(1),x(2),'.','color',col,'Markersize',20);
fx = func(x(1),x(2));
contour(xx,yy,ff,[fx,fx],'k','Linewidth',1);
% first do steepest decend step
a = linesearch(x,-g,g,func,eta,gam,jmax);
xnew = x - a*g;
gnew = gfun(xnew);
s(:,1) = xnew - x;
y(:,1) = gnew - g;
rho(1) = 1/(s(:,1)'*y(:,1));
plot([x(1),xnew(1)],[x(2),xnew(2)],'Linewidth',2,'color',col);
x = xnew;
g = gnew;
nor = norm(g);
plot(x(1),x(2),'.','color',col,'Markersize',20);
fx = func(x(1),x(2));
contour(xx,yy,ff,[fx,fx],'k','Linewidth',1);
iter = 1;
while nor > tol
    if iter < m
        I = 1 : iter;
        p = finddirection(g,s(:,I),y(:,I),rho(I));
    else
        p = finddirection(g,s,y,rho);
    end
    [a,j] = linesearch(x,p,g,func,eta,gam,jmax);
    if j == jmax
        p = -g;
        [a,j] = linesearch(x,p,g,func,eta,gam,jmax);
    end
    step = a*p;
    xnew = x + step;
    plot([x(1),xnew(1)],[x(2),xnew(2)],'Linewidth',2,'color',col);
```

```
    gnew = gfun(xnew);
    s = circshift(s,[0,1]);
    y = circshift(y,[0,1]);
    rho = circshift(rho,[0,1]);
    s(:,1) = step;
    y(:,1) = gnew - g;
    rho(1) = 1/(step'*y(:,1));
    x = xnew;
    g = gnew;
    fx = func(x(1),x(2));
    if nor > 1e-1
        contour(xx,yy,ff,[fx,fx],'k','Linewidth',1);
    end
    plot(x(1),x(2),'.','color',col,'Markersize',20);
    nor = norm(g);
    iter = iter + 1;
end
fprintf('L-BFGS: %d iterations, norm(g) = %d\n',iter,nor);
set(gca,'Fontsize',16);
xlabel('x_1','Fontsize',16);
ylabel('x_2','Fontsize',16);
end

%%
function [a,j] = linesearch(x,p,g,func,eta,gam,jmax)
    a = 1;
    f0 = func(x(1),x(2));
    aux = eta*g'*p;
    for j = 0 : jmax
        xtry = x + a*p;
        f1 = func(xtry(1),xtry(2));
        if f1 < f0 + a*aux
            break;
        else
            a = a*gam;
        end
    end
end
```

## 5. Methods for nonlinear least-squares problem

Reference: Nocedal and Wright, "Numerical Optimization" [1], Chapter 10.

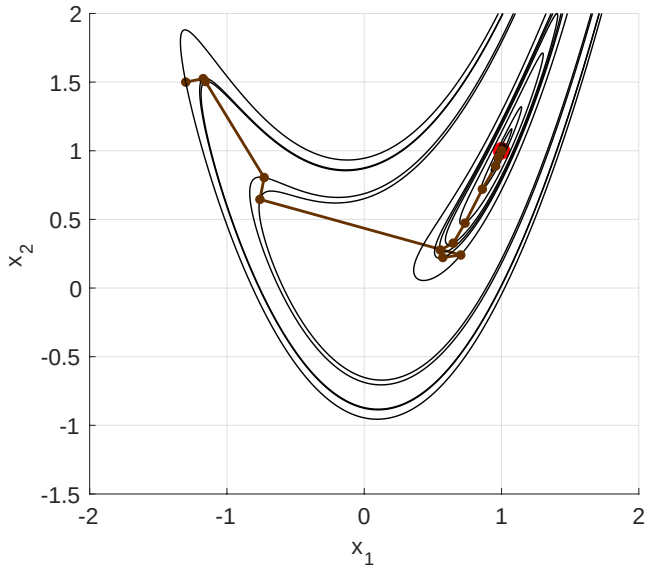


FIGURE 3. Iterates of L-BFGS applied to (7).

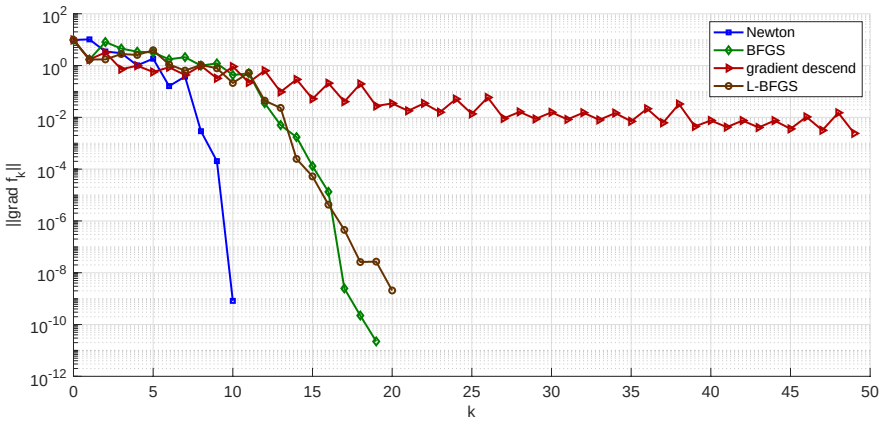


FIGURE 4. Decay of $\|\nabla f(\mathbf{x}_k)\|$ for various methods applied to (7).

In least-squares problems, the objective function has the following special form:

$$f(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{n} r_j^2(\mathbf{x}). \tag{10}$$

Objective functions of this form arise in many applications wherever there is a nonlinear model and experimental noisy data. The assumption that the noise is Gaussian leads to the objective function (10) [1] (Chapter 10). Loss functions in classification problems and training neural network problems can also be chosen of the form (10).

In this section, we will discuss two methods for solving nonlinear least squares problems: Gauss-Newton and Levenberg-Marquardt [1] (Chapter 10).

5.1. **The gradient and a handy approximation to the Hessian.** We will denote by $\mathbf{r}(\mathbf{x})$ and $J(\mathbf{x})$, respectively, the vector-function with components $r_j(\mathbf{x})$ in (10) and its Jacobian matrix:

$$\mathbf{r}(\mathbf{x}) := \begin{bmatrix} r_1(\mathbf{x}) \\ \vdots \\ r_n(\mathbf{x}) \end{bmatrix}, \quad J(\mathbf{x}) = \begin{bmatrix} \nabla r_1(\mathbf{x})^\top & \rightarrow \\ \vdots & \\ \nabla r_n(\mathbf{x})^\top & \rightarrow \end{bmatrix}. \tag{11}$$

Then $f$ and its gradient and Hessian are:

$$f(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2, \tag{12}$$

$$\nabla f(\mathbf{x}) = J(\mathbf{x})^\top \mathbf{r}(\mathbf{x}), \tag{13}$$

$$\nabla\nabla f(\mathbf{x}) = J(\mathbf{x})^\top J(\mathbf{x}) + \sum_{j=1}^{n} r_j(\mathbf{x})\nabla\nabla r_j(\mathbf{x}). \tag{14}$$

The second term in (14) can be small in two cases:

- if the residuals $r_j$ are small which is the case if the exact solution is zero of $\mathbf{r}$, and/or
- if $r_j$ are nearly linear in the neighborhood of the solution, i.e., $\nabla\nabla r_j$ are small.

Whether this is the case or not, both Gauss-Newton and Levenberg-Marquardt methods approximate the Hessian of $f$ with the first term in (14) only. Then Gauss-Newton follows the line-search strategy, while Levenberg-Marquardt employs the trust-region strategy.

5.2. **The Gauss-Newton method.** The Gauss-Newton method defines the search direction at step $k$ by solving

$$J_k^\top J_k \mathbf{p}_k = -J_k^\top \mathbf{r}_k, \tag{15}$$

where the subscript $k$ replaces the argument $\mathbf{x}_k$, and proceeds according to the standard routing for line-search methods. This choice of direction has several advantages.

- No computation of Hessians of $r_j$ is required.
- If $r_j$ are small or if $\nabla\nabla r_j$ are small, the Gauss-Netwon method converges almost as fast as Newton's method.
- If $J_k$ has linearly independent columns then the Gauss-Newton direction is a descent direction, i.e., $\mathbf{p}_k^\top J_k^\top \mathbf{r}_k < 0$. Indeed, from (15) we have:

$$\mathbf{p}_k^\top J_k^\top \mathbf{r}_k = -\mathbf{p}_k^\top J_k^\top J_k \mathbf{p}_k = -\|J_k \mathbf{p}_k\|^2 \leq 0. \tag{16}$$

If columns of $J_k$ are linearly independent, equality takes place if and only if $\mathbf{r}_k = 0$. Hence, if $\mathbf{x}_k$ is not the solution, we have strict inequality in (16) which means that $\mathbf{p}_k$ is a descent direction. Note that this is not true, in general, for the regular Newton's method unless it is applied to a strictly convex function.

- Equation (15) is the *normal equation* for the *linear least squares problem* $J_k\mathbf{p}_k = -\mathbf{r}_k$, i.e., its solution is the minimizer of

$$\min_{\mathbf{p}} \|J_k\mathbf{p} + \mathbf{r}_k\|^2. \tag{17}$$

This allows us to use methods for solving linear least squares problem such as QR decomposition via Householder reflections [2] for finding the search direction.

Convergence of the Gauss-Newton method to a stationary point under nonrestrictive conditions is guaranteed by the following

**Theorem 1.** *Suppose that all $r_j(\mathbf{x})$ are Lipschitz continuously differentiable in a neighborhood of the level set*

$$\{\mathbf{x} \mid f(\mathbf{x}) \le f(\mathbf{x}_0)\}$$

*and $J(\mathbf{x})$ satisfies the uniform full-rank condition*

$$\|J(\mathbf{x})\mathbf{z}\| \ge \gamma\|\mathbf{z}\| \quad \textit{for some} \quad \gamma > 0$$

*in this neighborhood. Then the iterates of the Gauss-Newton method with stepsizes satisfying the Wolfe conditions* [1] *(Section 3.1) converge to a stationary point of $f(\mathbf{x})$, i.e.,*

$$\lim_{k\to\infty} J_k^\top \mathbf{r}_k = 0.$$

The proof of this theorem follows from Theorem 3.2 in [1].

5.3. **The Levenberg-Marquardt method.** The Levenberg-Marquardt method follows the trust-region strategy. This means that at each step $k$, a trust region, typically of the form

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \equiv \|\mathbf{p}\| \le \Delta_k, \tag{18}$$

is given, an a constrained minimization problem for a quadratic model

$$m(\mathbf{p}) := f_k + \mathbf{p}^\top \mathbf{g}_k + \frac{1}{2}\mathbf{p}^\top B_k\mathbf{p} \tag{19}$$

is solved. Then the quality of the model is assessed by computing the ratio of the actual reduction to the expected reduction

$$\rho = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{p}_k)}{f(\mathbf{x}_k) - m(\mathbf{p}_k)} \tag{20}$$

and, depending on its value, the trust region radius for the next step is increased, left the same, or decreased. Finally, if $\rho$ is smaller than a user-prescribed threshold, the proposed step $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ is accepted or rejected. If the step is rejected, a new $\mathbf{p}_{k+1}$ is obtained at the next step as the solution to the constrained optimization problem in a smaller trust region. A pseudocode giving a template for any trust region method is outlined in Algorithm 1 I usually set $\eta = 0.1$ and $\Delta_0 = 0.2\Delta_{\max}$. There are several approaches to solving

**Algorithm 1:** Trust region template

**Input:**
Choose minimal and maximal radii of trust region $\Delta_{\min}$ and $\Delta_{\max}$;
Choose the initial trust region radius $\Delta_0 \in [\Delta_{\min}, \Delta_{\max}]$;
Choose threshold $\eta \in [0, 1/4)$ for accepting proposed step;
Choose initial approximation $\mathbf{x}_0$;
**for** $k = 1, 2, \ldots$ **do**
  Compute unconstrained minimizer $\mathbf{p}_k = -B_k^{-1}\mathbf{g}$ of $m(\mathbf{p})$;
  **if** $\|\mathbf{p}_k\| > \Delta_k$ **then**
    Solve constrained minimization problem (18)–(19) and get $\mathbf{p}_k$;
  **end**
  Compute the ratio $\rho$ (20);
  **if** $\rho < 1/4$ **then**
    Reduce the trust region radius: $\Delta_{k+1} = 0.25\Delta_k$;
  **else**
    **if** $\rho > 3/4$ *and* $\|\mathbf{p}_k\| = \Delta_k$ **then**
      Increase trust region radius $\Delta_{k+1} = \min(2\Delta_k, \Delta_{\max})$
    **end**
  **end**
  **if** $\rho_k > \eta$ **then**
    Accept step: $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$;
  **else**
    Reject step: $\mathbf{x}_{k+1} = \mathbf{x}_k$;
  **end**
**end**

the constrained minimization problem (18)–(19). The one used in Levenberg-Marquard gives the exact solution. Note that the quadratic model for Levenberg-Marquardt is of the form

$$m(\mathbf{p}) := \frac{1}{2}\|J_k\mathbf{p} + \mathbf{r}_k\|^2 = \frac{1}{2}\|\mathbf{r}_k\|^2 + \mathbf{p}^\top J_k^\top \mathbf{r}_k + \frac{1}{2}\mathbf{p}^\top J_k^\top J_k \mathbf{p}. \tag{21}$$

Therefore, the quadratic model is convex but it might be not strictly convex if $J_k$ has linearly dependent columns.

Recall Theorem 4.1 in [1] that characterizes the solution to the constrained minimization problem solved at each step of a trust-region method.

**Theorem 2.** *The vector* $\mathbf{p}^*$ *is a global solution to the problem*

$$\min_{\mathbf{p}\in\mathbb{R}^n} f + \mathbf{g}^\top\mathbf{p} + \frac{1}{2}\mathbf{p}^\top B\mathbf{p} \quad \textit{subject to} \quad \|\mathbf{p}\| \le \Delta, \tag{22}$$

*if and only if there exists* $\lambda \geq 0$ *such that*

$$(B + \lambda I)\mathbf{p}^* = -\mathbf{g}, \tag{23}$$

$$\lambda\,(\Delta - \|\mathbf{p}\|) = 0, \tag{24}$$

$$(B + \lambda I) \quad \textit{is positive semidefinite.} \tag{25}$$

Therefore, if the unconstrained minimizer $-B^{-1}\mathbf{g}$ lies outside the trust region, i.e, $\|B^{-1}\mathbf{g}\| > \Delta$, the solution to (23)–(24) lies on the trust region boundary, i.e., $\|\mathbf{p}\| = \Delta$. Let us discuss how to find $\mathbf{p}$ in this case. We have:

$$(B + \lambda I)\mathbf{p} = -\mathbf{g}, \quad \|\mathbf{p}\| = \Delta. \tag{26}$$

Since $B$ is symmetric and nonnegative definite as $B = J^\top J$, its spectral decomposition is

$$B = Q^\top \Lambda Q,$$

where $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_d]$ is orthogonal, i.e. its columns are orthonormal, and $\Lambda$ is diagonal. We always can order the eigenvalues in the nondecreasing order, i.e,

$$\Lambda = \mathrm{diag}\{\lambda_1, \ldots, \lambda_d\}, \quad \text{where} \quad 0 \leq \lambda_1 \leq \ldots \leq \lambda_d.$$

Using the spectral decomposition, the identity $\mathbf{p} = -(B + \lambda I)^{-1}\mathbf{g}$ can be rewritten as

$$\mathbf{p} = -\sum_{j=1}^{d} \frac{\mathbf{q}_j^\top \mathbf{g}}{\lambda_j + \lambda}\mathbf{q}_j. \tag{27}$$

Since columns of $Q$ are orthonormal, the identity $\|\mathbf{p}\| = \Delta$ becomes:

$$\sqrt{\sum_{j=1}^{d} \frac{(\mathbf{q}_j^\top \mathbf{g})^2}{(\lambda_j + \lambda)^2}} = \Delta. \tag{28}$$

Equation (28) is a 1D nonlinear equation that can be solved using Newton's method. Note that if $\lambda = 0$, then $\mathbf{p}$ is unconstrained minimizer with $\|\mathbf{p}\| > \Delta$ by our assumption. Hence, since all $\lambda_j$ are nonnegative, the solution $\lambda$ to (28) must be positive. On the other hand, since the left-hand side of (28) strictly decreases and tends to zero as $\lambda \to \infty$, we conclude that there exists a unique solution $\lambda^*$ to (28). Also, the difference between the left-and right-hand side of (28) behaves approximately as $C\lambda^{-1}$ while the difference between their reciprocals behaves approximately as a linear function which is beneficial for rapid convergence of Newton's iterations. So, we will solve numerically the equation

$$\phi(\lambda) := \frac{1}{\Delta} - \left[\sum_{j=1}^{d} \frac{(\mathbf{q}_j^\top \mathbf{g})^2}{(\lambda_j + \lambda)^2}\right]^{-1/2} = 0. \tag{29}$$

The Newton iteration is

$$\lambda^{(l+1)} = \lambda^{(l)} - \frac{\phi(\lambda^{(l)})}{\phi'(\lambda^{(l)})}. \tag{30}$$

The derivative of $\phi$ is given by:

$$\phi'(\lambda) = -\left[\sum_{j=1}^{d} \frac{(\mathbf{q}_j^\top \mathbf{g})^2}{(\lambda_j+\lambda)^2}\right]^{-3/2} \left[\sum_{j=1}^{d} \frac{(\mathbf{q}_j^\top \mathbf{g})^2}{(\lambda_j+\lambda)^3}\right]. \tag{31}$$

Left $\mathbf{p}_l$ be the solution to $(B+\lambda^{(l)})\mathbf{p} = -\mathbf{g}$. Then $\phi(\lambda^{(l)}) = \Delta^{-1} - \|\mathbf{p}_l\|^{-1}$, and the first factor in (31) is $\|\mathbf{p}_l\|^{-3}$. The second factor in (31) is the squared norm of the solution to $(B+\lambda I)^{3/2}\mathbf{q} = -\mathbf{g}$. Hence

$$\phi'(\lambda) = -\frac{\|\mathbf{q}\|^2}{\|\mathbf{p}_l\|^3}. \tag{32}$$

As a result, the Newton's update formula becomes:

$$\lambda^{(l+1)} = \lambda^{(l)} + \left[\frac{1}{\Delta} - \frac{1}{\|\mathbf{p}_l\|}\right]\frac{\|\mathbf{p}_l\|^3}{\|\mathbf{q}\|^2} = \lambda^{(l)} + \frac{\|\mathbf{p}_l\|^2}{\|\mathbf{q}\|^2}\left[\frac{\|\mathbf{p}_l\|-\Delta}{\Delta}\right]. \tag{33}$$

These considerations lead to the following subroutine for computing the solution to the constrained minimization problem in Levenberg-Marquardt.

```
% do Tikhonov regularization for the case J is rank-deficient
B = J'*J + (1e-12)*I;
pstar = -B\g; % unconstrained minimizer
if norm(pstar) <= R
    p = pstar;
else % solve constrained minimization problem
    lam = 1; % initial guess for lambda
    while 1
        B1 = B + lam*I;
        C = chol(B1); % do Cholesky factorization of B
        p = -C\(C'\g);  % solve B1*p = -g
        np = norm(p);
        dd = abs(np - R); % R is the trust region radius
        if dd < 1e-6
            break
        end
        q = C'\p; % solve C^\top q = p
        nq = norm(q);
        lamnew = lam + (np/nq)^2*(np - R)/R;
        if lamnew < 0
            lam = 0.5*lam;
        else
            lam = lamnew;
        end
    end
end
```

This Matlab routine does not take advantage of the fact that $(B+\lambda I)\mathbf{p} = -\mathbf{g}$ is really the normal equation for the linear least squares problem

$$\min_{\mathbf{p}} \frac{1}{2} \left\| \begin{bmatrix} J \\ \sqrt{\lambda} I \end{bmatrix} \mathbf{p} + \begin{bmatrix} \mathbf{g} \\ 0 \end{bmatrix} \right\|^2 .$$

To take an advantage of it, one needs to use low-level language. Then $(B + \lambda I)\mathbf{p} = -\mathbf{g}$ can be solved by QR decomposition implemented via a clever combination of Householder reflections and Givens rotations (see [1] (Section 10.3) and [2]).

For convergence theorems for Levenberg-Marquard consult Section 4 in [1]. There are many nuances, but in short, iterates of Levenberg-Marquardt converge to a stationary point provided that certain nonrestrictive conditions hold.

## 6. Stochastic gradient descent

The key reference for this section is Bottou, Curtis, and Nocedal, "Optimization Methods for Large-Scale Machine Learning", SIAM Review, 60, 2, 223–311, 2018 [3].

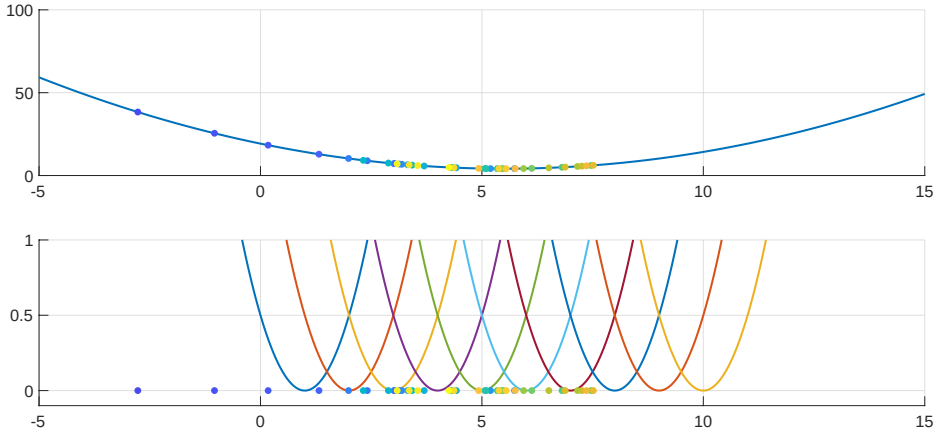


Figure 5. An example illustrating how stochastic gradient descent with a constant step length first rushes to the region where the minima of the individual functions are and then bounces around forever. Here, $f(x) = {}^1\!/_{20} \sum_{i=1}^{10} (x-i)^2$, step length is 0.3, batch size is 1. The iteration numbers are indicated by the `parula` colormap going from blue to yellow.

*Stochastic gradient descent (SG)* originates from the work by Robbins and Monroe (1951)"A stochastic approximation method" [4]. SG is designed for minimizing functions of the form

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} f_i(\mathbf{x}), \quad N \text{ is large.} \tag{34}$$

One simple version of SG runs as follows. One picks a *batch*, a random subset of indices $\mathcal{S}_k \subset \{1, \ldots, N\}$ and makes a step:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \left[ \frac{1}{|\mathcal{S}_k|} \sum_{j \in \mathcal{S}_k} \nabla f_j(\mathbf{x}_k) \right]. \tag{35}$$

A 1D example with batch size 1 is shown in Fig. 5. The expression in the square brackets is a stochastic approximation to $\nabla f(\mathbf{x}_k)$. Why is this a reasonable approximation? Often, there are many samples $\mathbf{x}_i$ in the training set (each sample $\mathbf{x}_i$ defines the corresponding $f_i$), and these samples can be split into groups consisting of similar samples. In this case, using just a subset of samples for estimating the gradient will give almost as good result but will be cheaper by the factor $m/N$. Moreover, typically, the dimensionality of machine learning optimization problems is very large. Hence, evaluating all $N$ gradients may cause a computer memory problem.

6.1. **Expected decrease of $f$ under SG iterations: assumptions and basic lemmas.** The SG algorithm offers a lot of flexibility for choosing batch sizes $|\mathcal{S}_k|$ and stepsizes $\alpha_k$. These can be fixed or variable and chosen according to some strategy enhancing performance. Moreover, the choice of the recipe for generating the direction for the step gives an additional flexibility. For example, one can calculate the direction of a step from scratch at each step, or incorporate the previously used directions, or even build up a stochastic estimate for the inverse Hessian and make the method a stochastic quasi-Newton.

We will denote the stochastic vector in the direction opposite to the direction of step $k$ by $g(\mathbf{x}_k; \xi_k)$ where $\xi_k$ is a random variable (generally, a vector random variable). For a simple SG with batch size 1,

$$\mathbf{g}(\mathbf{x}_k; \xi_k) = \nabla f_{\xi_k}(\mathbf{x}_k).$$

For SG with batch size $n_k$ we have

$$\mathbf{g}(\mathbf{x}_k; \xi_k) = \frac{1}{n_k} \sum_{i=1}^{n_k} \nabla f_{\xi_k(i)}(\mathbf{x}_k).$$

For a stochastic quasi-Newton version,

$$\mathbf{g}(\mathbf{x}_k; \xi_k) = H_k \frac{1}{n_k} \sum_{i=1}^{n_k} \nabla f_{\xi_k(i)}(\mathbf{x}_k).$$

In broad strokes, the SG algorithm is outlined in Algorithm 2.

To analyze SG we need to make some assumptions on the niceness of the objective function $f$.

**Assumption 1.** *$f : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz-continuous with constant $L$, i.e.,*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L\|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, \tag{36}$$

*where $\|\cdot\|$ is the 2-norm.*

**Algorithm 2:** SG algorithm

**Initialization:** Choose an initial vector $\mathbf{x}_0$ .
**for** $k = 0, 1, 2, \ldots$ **do**
  Generate a realization of the random variable $\xi_k$;
  Compute a stochastic vector $\mathbf{g}(\mathbf{x}_k, \xi_k)$;
  Choose a stepsize $\alpha_k$;
  Set a new iterate as $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}(\mathbf{x}_k, \xi_k)$;
**end**

Using (36) we can obtain a quadratic bound for the growth of $f$:

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{y}) + \int_0^1 \frac{df}{d\alpha}(\mathbf{y} + \alpha(\mathbf{x} - \mathbf{y}))d\alpha = f(\mathbf{y}) + \int_0^1 \nabla f(\mathbf{y} + \alpha(\mathbf{x} - \mathbf{y}))^\top (\mathbf{x} - \mathbf{y})d\alpha \\ &= f(\mathbf{y}) + \int_0^1 \left(\nabla f(\mathbf{y}) + \nabla f(\mathbf{y} + \alpha(\mathbf{x} - \mathbf{y})) - \nabla f(\mathbf{y})\right)^\top (\mathbf{x} - \mathbf{y})d\alpha \\ &\leq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \int_0^1 L\alpha \|\mathbf{x} - \mathbf{y}\|^2 d\alpha. \end{aligned}$$

Performing integration in the last identity, we obtain:

$$\boxed{f(\mathbf{x}) \leq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{1}{2} L \|\mathbf{x} - \mathbf{y}\|^2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.} \tag{37}$$

This identity allows us to establish the following

**Lemma 1.** *Under Assumption 1, iterates of Algorithm 2 satisfy*

$$\boxed{\mathbb{E}_{\xi_k}[f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) \leq -\alpha_k \nabla f(\mathbf{x}_k)^\top \mathbb{E}_{\xi_k}[\mathbf{g}(\mathbf{x}_k, \xi_k)] + \frac{L\alpha_k^2}{2} \mathbb{E}_{\xi_k}[\|\mathbf{g}(\mathbf{x}_k, \xi_k)\|^2] \quad \forall k \in \mathbb{Z}_+.} \tag{38}$$

*Proof.* By (37), the iterates satisfy

$$\begin{aligned} f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) &\leq \nabla f(\mathbf{x}_k)^\top (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2} L \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 \\ &= -\alpha_k \nabla f(\mathbf{x}_k)^\top \mathbf{g}(\mathbf{x}_k, \xi_k) + \frac{L\alpha_k^2}{2} \|\mathbf{g}(\mathbf{x}_k, \xi_k)\|^2. \end{aligned}$$

Taking expectations with respect to $\xi_k$ and noting that $\mathbf{x}_k$ is independent of $\xi_k$, we obtain (38). □

A good news is that if $-\mathbb{E}_{\xi_k}[\mathbf{g}(\mathbf{x}_k, \xi_k)]$ is a descent direction for $f$, i.e.,

$$\nabla f(\mathbf{x}_k)^\top \mathbb{E}_{\xi_k}[\mathbf{g}(\mathbf{x}_k, \xi_k)] > 0,$$

then for sufficiently small step length $\alpha$ we expect that $f$ will decrease as a result the step $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}(\mathbf{x}_k, \xi_k)$. To ensure that the random directions guarantee some minimal decrease we state

**Assumption 2.** *There exists a constant $\mu > 0$ such that for all $k \in \mathbb{Z}_+$*

$$\nabla f(\mathbf{x}_k)^\top \mathbb{E}_{\xi_k}[\mathbf{g}(\mathbf{x}_k, \xi_k)] \geq \mu \|\nabla f(\mathbf{x}_k)\|^2. \tag{39}$$

A bad news is that there is a harmful effect of the second term that is always positive. In order to limit its effect, we make one more

**Assumption 3.** *There exist constants $M \geq 0$ and $M_G \geq \mu^2 > 0$ ($\mu$ is from (39)) such that for all $k \in \mathbb{Z}_+$*

$$\mathbb{E}_{\xi_k}[\|\mathbf{g}(\mathbf{x}_k, \xi_k)\|^2] \leq M + M_G \|\nabla f(\mathbf{x}_k)\|^2. \tag{40}$$

**Example** Let us see how Assumptions 2 and 3 apply to the 1D example in Fig. 5 where

$$f(x) = \frac{1}{20} \sum_{i=1}^{10} (x - i)^2 \quad \text{and} \quad g = x - i \quad \text{where} \quad i = \texttt{randi}(10).$$

Let us start with Assumption 2. We have: $\nabla f(x) = x - 5.5$. The distribution of $g(x^*)$ is: each of the values $x - 1$, $x - 2$, …, $x - 9$, $x - 10$ is taken with the same probability 0.1. Hence $\mathbb{E}_i[g(x, i)] = x - 5.5$ as well, hence $\mu = 1$ works.

The stationary point is $x^* = 5.5$. The distribution of $g$ is: each of the values $-4.5$, $-3.5$, …, 3.5, 4.5 is taken with the same probability 0.1. Hence, the expectation of $g^2$ is

$$\mathbb{E}[g(x^*, i)^2] = 0.1 \cdot 2(4.5^2 + 3.5^2 + 2.5^2 + 1.5^2 + 0.5^2) = 8.25.$$

Hence, $M = 8.25$. Now, $\nabla f(x) = x - 5.5$ and $\|\nabla f(x)\|^2 = x^2 - 11x + 30.25$, while $g(x, i) = x - i$ and

$$\mathbb{E}[\|g(x, i)\|^2] = 0.1 \sum_{i=1}^{10} [x^2 - 2ix + i^2] = x^2 - 11x + 38.5.$$

Plugging this into (40) we get:

$$x^2 - 11x + 38.5. \leq 8.25 + M_G(x^2 - 11x + 30.25).$$

Hence, it suffices to pick $M_G = 1$. Note that $M_G = \mu^2$, hence the requirement that $M_G \geq \mu^2$ is satisfied.

Assumptions 2 and 3 allow us further elaborate the expected decrease of $f$ under SG iterations.

**Lemma 2.** *Under Assumptions 1, 2, and 3, the iterates of SG satisfy*

$$\boxed{\mathbb{E}_{\xi_k}[f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) \leq -\left(\mu - \frac{1}{2}\alpha_k L M_G\right) \alpha_k \|\nabla f(\mathbf{x}_k)\|^2 + \frac{1}{2}\alpha_k^2 L M.} \tag{41}$$

*Proof.* Plugging (39) and (40) to (38), we git:

$$\begin{aligned}
\mathbb{E}_{\xi_k}[f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) &\leq -\mu\alpha_k \|\nabla f(\mathbf{x}_k)\|^2 + \frac{1}{2}\alpha_k^2 L \left(M + M_G \|\nabla f(\mathbf{x}_k)\|^2\right) \\
&= -\left(\mu - \frac{1}{2}\alpha_k L M_G\right) \alpha_k \|\nabla f(\mathbf{x}_k)\|^2 + \frac{1}{2}\alpha_k^2 L M
\end{aligned}$$

as desired. □

Note that if $\alpha_k$ is small enough, the first term in the right-hand side of (41) is negative. The second term is always positive. These considerations are crucial for designing SG methods.

### 6.2. **Convergence of SG for strongly convex objective functions.**

**Assumption 4.** *The objective function $f : \mathbb{R}^d \to \mathbb{R}$ is strongly convex, i.e., there exists a constant $c > 0$ such that*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{c}{2}\|\mathbf{y} - \mathbf{x}\|^2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{42}$$

*Hence, $f$ has a unique minimizer $\mathbf{x}^* \in \mathbb{R}^d$ with $f^* := f(\mathbf{x}^*)$.*

Assumption 4 guarantees that $f$ grows away from its minimizer $\mathbf{x}^*$ at least as fast as the convex quadratic function $\frac{c}{2}\|\mathbf{x} - \mathbf{x}^*\|^2$. Note that the requirement of strong convexity is stronger than the one of strict convexity. If $f$ is twice continuously differentiable, strong convexity means that its Hessian is positive definite everywhere, and its eigenvalues are bounded from below by $c > 0$, while positive definiteness of the Hessian only will suffice to guarantee strict convexity. For example, the function $y = \sqrt{x^2 + 1}$ whose graph is the hyperbola lying above its two slant asymptotes $y = \pm x$ is strictly convex but not strongly convex. Its second derivative $y'' = (1 + x^2)^{-3/2}$ is positive everywhere but is approaches 0 as $|x| \to \infty$.

Comparing (42) and (37) we observe that $c \leq L$ ($L$ is the Lipschitz constant for $\nabla f$).

Assumption 4 also allows us to bound so called *optimality gap*.

**Proposition 1.** *Let $f$ satisfy Assumption 4. Then*

$$2c\,(f(\mathbf{x}) - f^*) \leq \|\nabla f(\mathbf{x})\|^2 \quad \forall \mathbf{x} \in \mathbb{R}^d. \tag{43}$$

*Proof.* Let us fix $\mathbf{x}$ and consider the quadratic model

$$q(\mathbf{y}) := f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{c}{2}\|\mathbf{y} - \mathbf{x}\|^2.$$

$q(\mathbf{y})$ has the unique minimizer $\hat{\mathbf{y}} := \mathbf{x} - \frac{1}{c}\nabla f(\mathbf{x})$ with

$$q(\hat{\mathbf{y}}) = f(\mathbf{x}) - \frac{1}{2c}\|\nabla f(\mathbf{x})\|^2.$$

Therefore, setting $\mathbf{y} = \mathbf{x}^*$ in (42), for any $\mathbf{x} \in \mathbb{R}^d$ we have:

$$f^* \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{x}^* - \mathbf{x}) + \frac{c}{2}\|\mathbf{x}^* - \mathbf{x}\|^2 \geq f(\mathbf{x}) - \frac{1}{2c}\|\nabla f(\mathbf{x})\|^2.$$

Hence, (43) readily follows. □

6.2.1. *Fixed stepsize.* Now we are ready to establish the first convergence result for SG with fixed stepsize for a strongly convex objective function. It is clear from (41) that the iterates will not be able to converge to the minimizer but will bounce around in its neighborhood as the first term in (41) tends to zero as we approach $\mathbf{x}^*$ while the second term remains constant. We will denote the *total expectation* of $f(\mathbf{x}_k)$ for any $k \in \mathbb{Z}_+$ by

$$\mathbb{E}[f(\mathbf{x}_k)] := \mathbb{E}_{\xi_0}\mathbb{E}_{\xi_1}\mathbb{E}_{\xi_2}\dots\mathbb{E}_{\xi_{k-1}}[f(\mathbf{x}_k)].$$

**Theorem 3.** *Under Assumptions 1, 2, 3, and 4, suppose that the SG method (Algorithm 2) is run with a fixed stepsize $\alpha$ satisfying*

$$0 < \alpha \leq \frac{\mu}{LM_G}. \tag{44}$$

*Then the expected optimality gap satisfies the following inequality for all $k \in \mathbb{Z}_+$:*

$$\boxed{\mathbb{E}[f(\mathbf{x}_k) - f^*] \leq \frac{\alpha LM}{2c\mu} + (1-\alpha c\mu)^k\left(f(\mathbf{x}_0) - f^* - \frac{\alpha LM}{2c\mu}\right) \to \frac{\alpha LM}{2c\mu}.} \tag{45}$$

Roughly speaking, this theorem says that the SG iterates with a fixed stepsize will reach a certain neighborhood of the optimal point and bounce there forever provided that the stepsize is not too large.

*Proof.* Plugging the bound (44) on the stepsize to (41) (the result of Lemma 2) and then inserting the optimality gap (43) we get:

$$\begin{aligned}
\mathbb{E}_{\xi_k}[f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) &\leq -\left(\mu - \frac{1}{2}\alpha LM_G\right)\alpha\|\nabla f(\mathbf{x}_k)\|^2 + \frac{1}{2}\alpha^2 LM \\
&\leq -\frac{\alpha\mu}{2}\|\nabla f(\mathbf{x}_k)\|^2 + \frac{1}{2}\alpha^2 LM \\
&\leq -\alpha c\mu(f(\mathbf{x}_k) - f^*) + \frac{1}{2}\alpha^2 LM.
\end{aligned}$$

Subtracting $f^*$ from both sides, rearranging terms, and taking total expectations, we get:

$$\mathbb{E}[f(\mathbf{x}_{k+1}) - f^*] \leq (1-\alpha c\mu)\mathbb{E}[f(\mathbf{x}_k) - f^*] + \frac{1}{2}\alpha^2 LM.$$

Subtracting $\frac{\alpha LM}{2\mu c}$ from both sides we get:

$$\mathbb{E}[f(\mathbf{x}_{k+1}) - f^*] - \frac{\alpha LM}{2\mu c} \leq (1-\alpha c\mu)\left(\mathbb{E}[f(\mathbf{x}_k) - f^*] - \frac{\alpha LM}{2\mu c}\right). \tag{46}$$

Note that

$$0 < \alpha c\mu \leq \frac{c\mu^2}{LM_G} \leq \frac{c\mu^2}{L\mu^2} = \frac{c}{L} \leq 1.$$

Here we used the assumption that $M_G \geq \mu^2$(see Assumption 3) you might have been wondering about what is it for. Therefore,

$$0 \leq (1-\alpha c\mu) < 1.$$

Applying (46) repeatedly we obtain (45). This completes the proof. □

6.2.2. *Decreasing step size.* As we have proven, SG with a fixed step size does not converge to the minimizer. In order to achieve convergence, we need to reduce stepsize as we progress but not too fast: a condition for stepsizes $\alpha_k$ proven in [4] is

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty. \tag{47}$$

Note that $\alpha_k \sim k^{-1}$ for large $k$ satisfies (47). Moreover, if we want to reduce stepsize by a factor of 2, we need to do it on a certain schedule. One such a schedule is: do $m_0$ steps of size $\alpha$, then do $m_1 = m_0$ steps of size $2^{-1}\alpha$, ..., then do $m_k = m_0 2^k/k$ steps of size $2^{-k}\alpha$, and so on. Then

$$\sum_{k=1}^{\infty} m_0 \frac{2^k}{k} \frac{\alpha}{2^k} = \infty, \quad \sum_{k=1}^{\infty} m_0 \frac{2^k}{k} \frac{\alpha^2}{2^{2k}} = \sum_{k=1}^{\infty} m_0 \frac{\alpha^2}{k 2^k} < \infty.$$

The following theorem offers another schedule for stepsize reduction for which error decay goes as $O(1/k)$.

**Theorem 4.** *Under Assumptions 1, 2, 3,and 4, suppose that the SG Algorithm (Algorithm 2) is run with a stepsize sequence $\alpha_k$, $k \in \mathbb{N}$, such that*

$$\alpha_k = \frac{\beta}{\gamma + k} \quad \textit{for some} \quad \beta > \frac{1}{c\mu} \quad \text{and} \quad \gamma > 0 \quad \textit{such that} \quad \alpha_1 \leq \frac{\mu}{LM_G}. \tag{48}$$

*Then for all $k \in \mathbb{N}$, the expected optimality gap satisfies*

$$\boxed{\mathbb{E}[f(\mathbf{x}_k) - f^*] \leq \frac{\nu}{\gamma + k},} \tag{49}$$

*where*

$$\nu := \max\left\{ \frac{\beta^2 LM}{2(\beta c\mu - 1)}, (\gamma + 1)(f(\mathbf{x}_1) - f^*) + \frac{1}{2}\frac{\gamma M \mu^2}{LM_G^2} \right\}. \tag{50}$$

*Proof.* Repeating the start of the proof of Theorem 3 we obtain the inequality:

$$\mathbb{E}_{\xi_k}[f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) \leq -\alpha_k c\mu (f(\mathbf{x}_k) - f^*) + \frac{1}{2}\alpha_k^2 LM$$

Subtracting $f^*$ from both sides, rearranging terms, and taking total expectations, we get:

$$\mathbb{E}[f(\mathbf{x}_{k+1}) - f^*] \leq (1 - \alpha_k c\mu)\mathbb{E}[f(\mathbf{x}_k) - f^*] + \frac{1}{2}\alpha_k^2 LM.$$

Now we proceed by induction. It follows from definition of $\nu$ that (49) holds for $k = 1$. Indeed,

$$\frac{\nu}{\gamma + 1} \geq f(\mathbf{x}_1) - f^*.$$

Induction assumption: (49) holds for $k$:

$$\mathbb{E}[f(\mathbf{x}_k) - f^*] \leq \frac{\nu}{\gamma + k}.$$

Induction step:

$$\begin{aligned}
\mathbb{E}[f(\mathbf{x}_{k+1}) - f^*] &\le (1 - \alpha_k c\mu)\mathbb{E}[f(\mathbf{x}_k) - f^*] + \frac{1}{2}\alpha_k^2 LM \\
&= \left(1 - \frac{\beta c\mu}{\gamma + k}\right)\frac{\nu}{\gamma + k} + \frac{1}{2}\frac{\beta^2}{(\gamma + k)^2}LM \\
&= \frac{\gamma + k - 1}{(\gamma + k)^2}\nu \underbrace{- \frac{\beta c\mu - 1}{(\gamma + k)^2}\nu + \frac{1}{2}\frac{\beta^2}{(\gamma + k)^2}LM}_{\le 0 \text{ by definition of } \nu} \\
&\le \frac{\gamma + k - 1}{(\gamma + k)^2}\nu \\
&< \frac{\nu}{\gamma + k + 1} \quad \text{as} \quad (\gamma + k)^2 > (\gamma + k + 1)(\gamma + k - 1).
\end{aligned}$$

This completes the proof. □

**Example** Fig. 6 shows an example of application of the SG algorithm to

$$f(x) = \frac{1}{2n}\sum_{i=1}^{n}(x - i)^2 \tag{51}$$

with $n = 10$, initial guess $x_0 = -5$, batch size 1, and stepsizes reduced according to the following schedule:

```
step = 0.3;
N = 15;
NN = 10;
for ii = 1 : NN
    s = step/2^ii;
    nsteps = ceil(N*2^ii/ii);
    for i = 1 : nsteps
        k = randi(n);
        g = grad(k,x);
        x = x - g*s;
    end
end
```

Then the expectations $\mathbb{E}[f(x_k) - f^*]$ is estimated as the average over $10^5$ runs of this algorithm.

6.3. **SG for nonconvex objective functions.** If the objective function is not convex, then, under the rest of assumptions that we have made for the strongly convex case, there is a subsequence of iterates at which the expected norm squared of the gradient approaches zero, i.e., a subsequence of iterates approaches a stationary point. If we amplify smoothness requirements for $f$, the iterates will approach a stationary point. This is summarized in the following

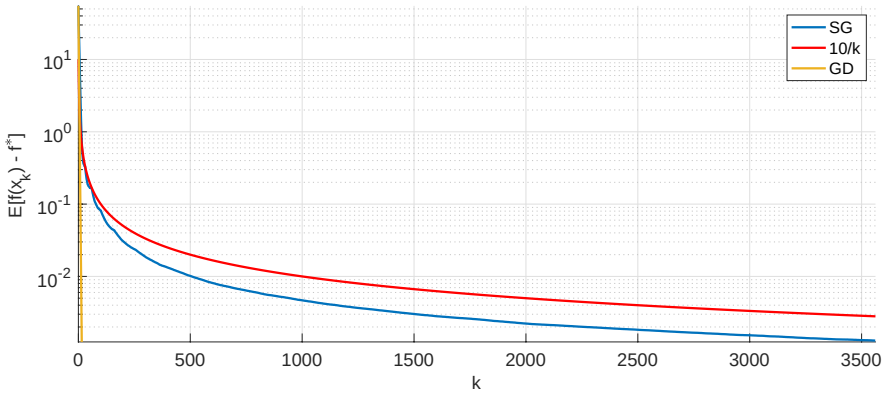


FIGURE 6. Decay of the optimality gap for SG applied to (51) with $n = 10$ and batch size 1. The graph of $10/k$ is shown in red for comparison. The regular gradient descent reduces the optimality gap to the same value in 16 iterations (see the yellow plot).

**Theorem 5.** *Under Assumptions 1, 2, and 3, suppose that the SG algorithm is run with stepsizes satisfying (47). Then*

$$\lim_{k\to\infty} \inf \mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2] = 0. \tag{52}$$

*If, in addition $f$ is twice continuously differentiable, and the mapping $\mathbf{x} \mapsto \|\nabla f(\mathbf{x})\|^2$ has Lipschitz-continuous derivatives, then*

$$\lim_{k\to\infty} \mathbb{E}[\|\nabla f(\mathbf{x}_k)\|^2] = 0. \tag{53}$$

I am referring an interested reader to [3] (Section 4.3) to look up the proof.

## 7. Gradient descent and accelerated gradient descent

In this section, we will explore the convergence of gradient descent and accelerated gradient descent with constant stepsize. We will discuss several variants of accelerated gradient descent, in particular, Nesterov accelerated gradient and ADAM. I am using the following sources:

- Convergence of gradient descent: lecture by Ryan Tibshirani (Carnegie Mellon University): http://www.stat.cmu.edu/ ryantibs/convexopt-F13/scribes/lec6.pdf;
- The original paper by Yu. Nesterov (1983) in Russian;
- Convergence of Nesterov Accelerated Gradient: slides by Andersen Ang (UMONS, Belgium): https://angms.science/doc/CVX/CVX_NAGD.pdf;
- A blog by Sebastien Ruder giving a good overview for modern methods for large-scale optimization: https://ruder.io/optimizing-gradient-descent/;

- A paper by A. Botev, G. Lever, and D. Barber, (CS, University College, London) "Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent" that gives a nice perspective on Nesterov's algorithm: https://arxiv.org/pdf/1607.01981.pdf;
- A paper by D. P. Kingma and J. L. Ba "Adam: A Method for Stochastic Optimization" where ADAM is introduced: https://arxiv.org/pdf/1412.6980.pdf.

7.1. **Convergence analysis for gradient descent with constant learning rate.** We have the convergence result given as Theorem 4 for stochastic gradient descent (SG) for strongly convex functions with Lipschitz-continuous gradient. It shows that the optimality gap of SG converges as $O(k^{-1})$ provided that the stepsize is reduced harmonically, $\alpha_k = \beta(k+\gamma^{-1})$ with $\gamma$ and $\beta$ chosen appropriately. Noting that $M = 0$, and $\mu = M_G = 1$ for the gradient descent (GD), we easily can adapt Theorem 3 for GD with constant stepsize $\alpha \in (0, 1/L]$ and conclude that

$$f(\mathbf{x}_k) - f^* \leq (1-\alpha c)^k (f(\mathbf{x}_1) - f^*),$$

where the parameter $c > 0$ characterizes the strong convexity of $f$. Now we will prove another convergence result for GD characterizing its convergence rate as $O(1/k)$. Note that strong convexity is no longer required.

**Theorem 6.** *Suppose the function $f : \mathbb{R}^d \to \mathbb{R}$ is convex and its gradient is Lipschitz continuous with constant $L > 0$. Then if we run gradient descent for $k$ iterations with a fixed step size $0 < \alpha \leq 1/L$, it will yield a solution $f(\mathbf{x}_k)$ which satisfies*

$$f(\mathbf{x}_k) - f^* \leq \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{2\alpha k}. \tag{54}$$

*where $f^* \equiv f(\mathbf{x}^*)$ is the minimum of $f$.*

*Proof.* The iterations of GD are:

$$x_{k+1} = x_k - \alpha \nabla f_k, \quad \text{where} \quad f_k \equiv f(\mathbf{x}_k).$$

Recall that the Lipschitz continuity of $\nabla f$ implies (see (37)) that

$$\begin{aligned} f(\mathbf{x}_{k+1}) = f(x_k - \alpha \nabla f_k) &\leq f(\mathbf{x}_k) - \alpha \|\nabla f_k\|^2 + \frac{\alpha^2 L}{2} \|\nabla f_k\|^2 \\ &= f(\mathbf{x}_k) - \alpha \left(1 - \frac{\alpha L}{2}\right) \|\nabla f_k\|^2. \end{aligned}$$

Since $0 < \alpha \leq 1/L$, we have $0 < \alpha L/2 \leq 1/2$. Therefore,

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \frac{\alpha}{2} \|\nabla f_k\|^2. \tag{55}$$

To obtain further bounds, we need to use the convexity of $f$. We have:

$$f(\mathbf{x}) \leq f^* + \nabla f(\mathbf{x})^\top (\mathbf{x} - \mathbf{x}^*) \quad \forall \mathbf{x} \in \mathbb{R}^d. \tag{56}$$

Subtracting $f^*$ from both sides of (55) and using (56) then we get:

$$\begin{aligned} f(\mathbf{x}_{k+1}) - f^* &\leq f(\mathbf{x}_k) - f^* - \frac{\alpha}{2}\|\nabla f_k\|^2 \\ &\leq \nabla f_k^\top (\mathbf{x}_k - \mathbf{x}^*) - \frac{\alpha}{2}\|\nabla f_k\|^2 \\ &= \frac{1}{2\alpha}\left(2\alpha \nabla f_k^\top (\mathbf{x}_k - \mathbf{x}^*) - \alpha^2\|\nabla f_k\|^2 - \|\mathbf{x}_k - \mathbf{x}^*\|^2 + \|\mathbf{x}_k - \mathbf{x}^*\|^2\right) \\ &= \frac{1}{2\alpha}\left(\|\mathbf{x}_k - \mathbf{x}^*\|^2 - \|\mathbf{x}_k - \mathbf{x}^* - \alpha\nabla f_k\|^2\right) \\ &= \frac{1}{2\alpha}\left(\|\mathbf{x}_k - \mathbf{x}^*\|^2 - \|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2\right). \end{aligned}$$

Summing the last inequality from $j = 1$ till $j = k$ we obtain:

$$\begin{aligned} \sum_{j=1}^{k} (f(\mathbf{x}_j) - f^*) &\leq \frac{1}{2\alpha}\sum_{j=1}^{k}\left(\|\mathbf{x}_{j-1} - \mathbf{x}^*\|^2 - \|\mathbf{x}_j - \mathbf{x}^*\|^2\right) \\ &= \frac{1}{2\alpha}\left(\|\mathbf{x}_0 - \mathbf{x}^*\|^2 - \|\mathbf{x}_k - \mathbf{x}^*\|^2\right) \\ &\leq \frac{1}{2\alpha}\|\mathbf{x}_0 - \mathbf{x}^*\|^2. \end{aligned}$$

Finally, we use the fact that $f$ is non-increasing at any iteration which means that

$$f(\mathbf{x}^k) - f^* \leq \frac{1}{k}\sum_{j=1}^{k}(f(\mathbf{x}_j) - f^*) \leq \frac{1}{2k\alpha}\|\mathbf{x}_0 - \mathbf{x}^*\|^2,$$

as desired. □

7.2. **Nesterov's accelerated gradient: motivation.** We start the discussion on Nesterov's accelerated gradient (NAG) with an intuitive perspective on it given in [5]. Thinking of Newton's mechanics for a body of mass $m$ being acted upon by the potential force $\nabla f$ and experiencing friction proportional to its speed, we have:

$$\dot{\mathbf{v}} = -\gamma\mathbf{v} - \frac{1}{m}\nabla f(\mathbf{x}), \tag{57}$$

$$\dot{\mathbf{x}} = \mathbf{v}. \tag{58}$$

Discretizing this equation in time with timestep 1, we obtain

$$\mathbf{v}_{k+1} = (1-\gamma)\,\mathbf{v}_k - \frac{1}{m}\nabla f(\mathbf{x}_k), \tag{59}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1}. \tag{60}$$

We will refer to this algorithm as gradient descent with momentum and abbreviate it as MOM.

Now we consider NAG. It is given by:

$$\mathbf{y}_{k+1} = (1+\mu_k)\,\mathbf{x}_k - \mu_k \mathbf{x}_{k-1}, \tag{61}$$

$$\mathbf{x}_{k+1} = \mathbf{y}_{k+1} - \alpha_k \nabla f(\mathbf{y}_{k+1}), \tag{62}$$

where Nesterov proposed schedule $\mu_k = 1 - \frac{3}{5+k}$ and fixed $\alpha_k$. Introducing $\mathbf{v}_k := \mathbf{x}_k - \mathbf{x}_{k-1}$, we see that $\mathbf{y}_{k+1} = \mathbf{x}_k + \mu_k \mathbf{v}_k$ and hence

$$\mathbf{x}_k = \mathbf{y}_{k+1} - \mu_k \mathbf{v}_k. \tag{63}$$

Furthermore,

$$\mathbf{x}_{k+1} = \mathbf{y}_{k+1} - \alpha_k \nabla f(\mathbf{x}_k + \mu_k \mathbf{v}_k). \tag{64}$$

Subtracting (63) from (64) we obtain the iteration for $\mathbf{x}_k$ and $\mathbf{v}_k$:

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \nabla f(\mathbf{x}_k + \mu_k \mathbf{v}_k), \tag{65}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1}. \tag{66}$$

Comparing (65)–(66) with (59)–(60) we see that NAG evaluates the gradient at the extrapolated position $\mathbf{x}_k + \mu_k \mathbf{v}_k$.

7.3. **Convergence of Nesterov's accelerated descent.** The original version of NAG is a bit different from a different from (61)–(62). The Algorithm outlined below gives a slightly simplified modification of the original algorithm: the stepsize $\alpha$ is set to be $1/L$ rather than being chosen at each step by the bisection method.

---

**Initialization** *Choose the initial approximation* $\mathbf{y}_0$. *Set:* $k = 0$, $\lambda_0 = 0$, $\mathbf{x}_0 = \mathbf{y}_0$.
**for** *k = 0,1,2,...* **do**
Gradient update:

$$\mathbf{y}_{k+1} = \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k);$$

Extrapolation weight:

$$\lambda_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4\lambda_k^2}\right), \quad \gamma_k = \frac{1-\lambda_k}{\lambda_{k+1}};$$

Extrapolation:

$$\mathbf{x}_{k+1} = (1-\gamma_k)\mathbf{y}_{k+1} + \gamma_k \mathbf{y}_k;$$

**end**

---

**Theorem 7.** *Suppose the function* $f : \mathbb{R}^d \to \mathbb{R}$ *is convex and its gradient is Lipschitz continuous with constant* $L > 0$. *Then the iterates of NAG with a constant stepsize* $\alpha \equiv 1/L$ *converge to the optimal value* $f^*$ *at rate* $O(k^{-2})$ *as*

$$f(\mathbf{y}_k) - f^* \le \frac{2L\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{k^2}. \tag{67}$$

*where* $f^* \equiv f(\mathbf{x}^*)$ *is the minimum of* $f$.

The proof of this theorem is long and tricky. I refer an interested reader to Ang's slides.

None that NAG can be easily converted to a stochastic version by approximating the gradient of $f$ as we have done in SG.

7.4. **Adam.** See D. P. Kingma and J. L. Ba "Adam: A Method for Stochastic Optimization": https://arxiv.org/pdf/1412.6980.pdf. A motivation for Adam and its connections to other optimizers are discussed in S. Ruder's blog.

Adaptive Moment Estimation (Adam) computes adaptive learning rates for each component of $\mathbf{x}$. At each step, we compute the decaying averages for the gradient and elementwise gradient squared (the first and the second moments):

$$m_k = \beta_1 m_{k-1} + (1-\beta_1)\nabla f(\mathbf{x}_k), \quad v_k = \beta_2 v_{k-1} + (1-\beta_2)\left[\nabla f(\mathbf{x}_k) \odot \nabla f(\mathbf{x}_k)\right].$$

Since at the start of the algorithm, these moments are biased toward zero, as $\beta_1$ and $\beta_2$ are close to 1, these biases are counteracted by bias-corrected estimates:

$$\hat{m}_k = \frac{m_k}{1-\beta_1^k}, \quad \hat{v}_k = \frac{v_k}{1-\beta_2^k}.$$

The update for Adam is given by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\eta}{\sqrt{\hat{v}_k} + \epsilon}\hat{m}_k.$$

Default values for the parameters are:

$$\beta_1 = 0.9, \quad \beta_2 = 0.999, \quad \epsilon = 10^{-8}, \quad \eta = 0.001.$$

## 8. Basics of constrained optimization

Ref.: Nocedal and Wright [1], Chapter 12.

We will consider a general constrained optimization problem of the form

$$\begin{aligned} & f(\mathbf{x}) \to \min \quad \text{subject to} \\ & c_i(\mathbf{x}) = 0, \quad i \in \mathcal{E}, \quad \text{(equality constraints)} \\ & c_i(\mathbf{x}) \geq 0, \quad i \in \mathcal{I}, \quad \text{(inequality constraints)}. \end{aligned} \tag{68}$$

Any point $\mathbf{x}$ satisfying $c_i(\mathbf{x}) = 0$, $i \in \mathcal{E}$ and $c_i(\mathbf{x}) \geq 0$, $i \in \mathcal{I}$, is called *feasible*. We assume that $f$ and $c_i$, $i \in \mathcal{E} \cup \mathcal{I}$, are continuously differentiable.

In order to understand how to solve (68), first recall the method of Lagrange multipliers from your calculus course. A helpful quick reminder is given in the Wiki article "Lagrange Multiplier"–look at Figure 1 there.

Imagine that we have a single equality constraint $c(\mathbf{x}) = 0$. Then the minimizer of $f(\mathbf{x})$ subject to $c(\mathbf{x}) = 0$ lies at a point $\mathbf{x}^*$ such that the level set $\{\mathbf{x} \in \mathbb{R}^d \mid f(\mathbf{x}) = f(\mathbf{x}^*)\}$ is tangent to the hypersurface $c(\mathbf{x}) = 0$. Indeed, moving along the hypersurface $c(\mathbf{x}) = 0$ and keeping track of the values of $f(\mathbf{x})$, the extreme values of $f$ will be achieved at the points where the level set of $f$ is tangent to $c(\mathbf{x}) = 0$ (see Fig. 7(a)). At such points, the gradients of $f$ and $c$ will be parallel to each other, i.e., they will relate via

$$\nabla f(\mathbf{x}) = \lambda \nabla c(\mathbf{x}). \tag{69}$$

The factor $\lambda$ is called *the Lagrange multiplier*. The condition (69) motivates the definition of *the Lagrangian function*

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda c(\mathbf{x}). \tag{70}$$

Stationary points of $L(\mathbf{x}, \lambda)$ are those where its gradient is zero, i.e.,

$$\nabla_{\mathbf{x}} L = \nabla f(\mathbf{x}) - \lambda \nabla c(\mathbf{x}) = 0, \tag{71}$$

$$\nabla_{\lambda} L = -c(\mathbf{x}) = 0, \tag{72}$$

Hence, at every stationary point of $L(\mathbf{x}, \lambda)$ we have: $(i)$ $c(\mathbf{x}) = 0$ and $(ii)$ the gradients of $f$ and $c$ are parallel.

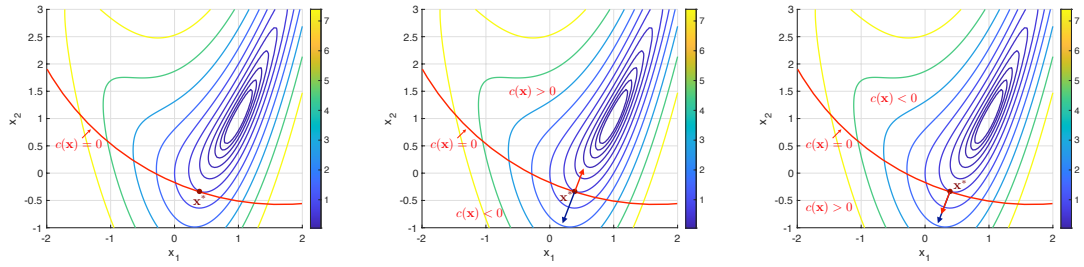


FIGURE 7. The geometric sense of Lagrange multiplier. (a): The solution to the equality-constrained minimization problem is reached at the point $\mathbf{x}^*$ where the level set of $f$ is tangent to $c(\mathbf{x}) = 0$. (b): The unconstrained minimum of $f$ is the solution to the inequality-constrained optimization problem $c(\mathbf{x}) \geq 0$. (c): At the constrained minimum of $f$, $\nabla f(\mathbf{x}^*) = \lambda \nabla c(\mathbf{x}^*)$, where $\lambda > 0$. *The level sets are those of the Rosenbrock function* $f(x_1, x_2) = (1 - x_1)^2 + (x_2 - x_1^2)^2$.

Now we replace the equality constraint $c(\mathbf{x}) = 0$ with the inequality constraint $c(\mathbf{x}) \geq 0$. For visuality, we assume that level sets of $f(\mathbf{x})$ are simple closed surfaces. This implies that $f$ has a unique local minimizer, and it is its global minimizer.

- If the minimum of $f$ lies in the region $c(\mathbf{x}) \geq 0$, the constrained minimum coincides with the unconstrained minimum. Apart from this global minimum, consider the point $\mathbf{x}^*$ where a level set of $f$ touches the level set $c(\mathbf{x}) = 0$. The gradients $\nabla f(\mathbf{x}^*)$ and $\lambda \nabla c(\mathbf{x}^*)$ are antiparallel, i.e., $\nabla f(\mathbf{x}^*) = \lambda \nabla c(\mathbf{x}^*)$ for some $\lambda < 0$ (see Fig. 7(b)).
- If the global minimizer of $f$ does not belong to the feasible set where $c(\mathbf{x}) \geq 0$ as in Fig. 7(c), the constrained minimum will be achieved at the point $\mathbf{x}^*$ lying at $c(\mathbf{x}) = 0$ where $\nabla f(\mathbf{x}^*) = \lambda \nabla c(\mathbf{x}^*)$ for some $\lambda > 0$. In this case, we say that the constraint $c(\mathbf{x}) \geq 0$ is *active* at the solution $\mathbf{x}^*$.

This illustrative example shows that the sign of the Lagrange multiplier is important and gives an intuition for the *Karush-Kuhn-Tucker* first-order optimality conditions.

8.1. **Karush-Kuhn-Tucker optimality conditions.** This section is based on J. Nocedal and S. Wright "Numerical Optimization" [1], Chapter 12.

**Definition 1.** *The active set* $\mathcal{A}(\mathbf{x})$ *at any feasible* $\mathbf{x}$ *consists of the equality constraint indices from* $\mathcal{E}$ *together with the indices of the inequality constraints* $i$ *for which* $c_i(\mathbf{x}) = 0$*; that is,*

$$\mathcal{A}(\mathbf{x}) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(\mathbf{x}) = 0\}. \tag{73}$$

At any feasible point $\mathbf{x}$, the inequality constraint $c_i(\mathbf{x}) \geq 0$ is *active* if $c_i(\mathbf{x}) = 0$ and inactive if $c_i(\mathbf{x}) > 0$.

The *Lagrangian function* is defined by

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(\mathbf{x}). \tag{74}$$

**Definition 2.** *A feasible point* $\mathbf{x}^*$ *is a local solution of* (68) *if all feasible sequences* $\mathbf{z}_k \to \mathbf{x}^*$ *as* $k \to \infty$ *have the property that* $f(\mathbf{z}_k) \geq f(\mathbf{x}^*)$ *for all* $k$ *sufficiently large.*

**Definition 3.** *A direction* $\mathbf{d}$ *at a feasible point* $\mathbf{x}$ *is feasible if* $\nabla c_i(\mathbf{x})^\top \mathbf{d} = 0$ $\forall i \in \mathcal{E}$ *and* $\nabla c_i(\mathbf{x})^\top \mathbf{d} \geq 0$ $\forall i \in \mathcal{A}(\mathbf{x}) \cap \mathcal{I}$.

**Definition 4.** *The vector* $\mathbf{d}$ *is a tangent vector to the feasible set* $\Omega$ *at a point* $\mathbf{x}$ *if there are a feasible sequence* $\mathbf{z}_k \to \mathbf{x}$ *and a sequence of positive scalars* $t_k \to 0$ *as* $k \to \infty$ *such that*

$$\lim_{k \to \infty} \frac{\mathbf{z}_k - \mathbf{x}}{t_k} = \mathbf{d}. \tag{75}$$

*The set of all tangent vectors at* $\mathbf{x}$ *is called the tangent cone and is denoted by* $T_\Omega(\mathbf{x})$.

The first-order optimality conditions known as the *Karush-Kuhn-Tucker (KKT) conditions* are stated in the following theorem.

**Theorem 8** (KKT with LICQ)**.** *Suppose* $\mathbf{x}^*$ *is a local solution of* (68) *where* $f$ *and* $c_i$*'s are continuously differentiable, and the set of gradients of active constraints*

$$\{\nabla c_i(\mathbf{x}^*),\ i \in \mathcal{A}(\mathbf{x}^*)\} \tag{76}$$

*is linearly independent. Then there is a Lagrange multiplier vector* $\boldsymbol{\lambda}^* = \{\lambda_i\}_{i \in (\mathcal{E} \cup \mathcal{I})}$ *such that the following conditions are satisfied at* $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$*:*

$$\nabla_{\mathbf{x}} L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0, \tag{77}$$

$$c_i(\mathbf{x}^*) = 0 \quad \forall i \in \mathcal{E}, \tag{78}$$

$$c_i(\mathbf{x}^*) \geq 0 \quad \forall i \in \mathcal{I}, \tag{79}$$

$$\lambda_i^* \geq 0 \quad \forall i \in \mathcal{I}, \tag{80}$$

$$\lambda_i^* c_i(\mathbf{x}^*) = 0 \quad \forall i \in \mathcal{E} \cup \mathcal{I}. \tag{81}$$

Condition (83) known as the *linear independence constraint qualification (LICQ)* eliminates the situations when equations (77)–(81) do not hold at the solution point $\mathbf{x}^*$ for any $\boldsymbol{\lambda}^*$. For example, consider the following problem:

$$
\begin{aligned}
& f(\mathbf{x}) = x_1^2 + (x_2+1)^2 \ \rightarrow \ \min \quad \text{subject to} \\
& c_1(\mathbf{x}) = x_2^3 - x_1 \geq 0 \\
& c_2(\mathbf{x}) = x_2^3 + x_1 \geq 0.
\end{aligned} \tag{82}
$$

The curves $c_1(\mathbf{x}) = 0$ and $c_2(\mathbf{x}) = 0$ (red) and level sets of $f(\mathbf{x})$ are shown in Fig. 8. The feasible set is shaded in light blue. The optimal point is $\mathbf{x}^* = [0,0]^\top$. At $\mathbf{x}^*$, we have:

$$
\nabla f(\mathbf{x}^*) = [0,-2]^\top, \quad \nabla c_1(\mathbf{x}^*) = [-1,0]^\top, \quad \nabla c_2(\mathbf{x}^*) = [1,0]^\top.
$$

It is easy to check that LICQ does not hold and (77) does not hold either.

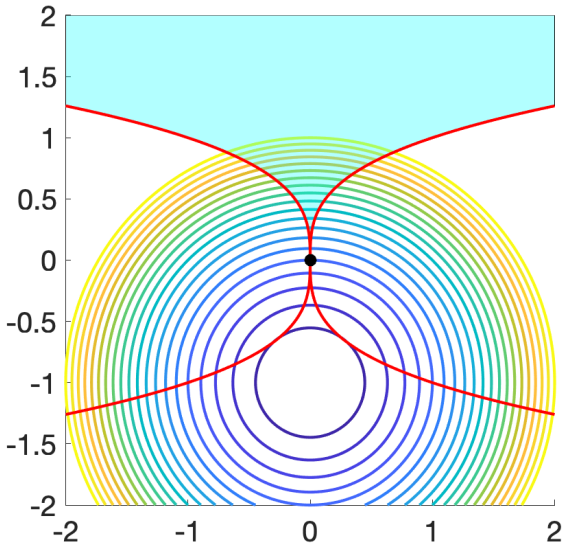


FIGURE 8. An example (82) showing that the KKT conditions might fail if LICQ does not hold.

LICQ is not the only condition under which the KKT conditions (77)–(81) hold at the optimal point. The KKT conditions also hold at the optimal point if the constraints are linear. In this case, their linear independence is not required.

**Theorem 9** (KKT with linear constraints)**.** *Suppose* $\mathbf{x}^*$ *is a local solution of* (68) *where* $f$ *is continuously differentiable and* $c_i$*'s are linear:*

$$
c_i(\mathbf{x}) = \mathbf{a}_i^\top \mathbf{x} - b_i, \quad i \in \mathcal{E} \cup \mathcal{I}. \tag{83}
$$

*Then there is a Lagrange multiplier vector* $\boldsymbol{\lambda}^* = \{\lambda_i\}_{i\in(\mathcal{E}\cup\mathcal{I})}$ *such that the following conditions are satisfied at* $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$*:*

$$\nabla_{\mathbf{x}} L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0, \tag{84}$$

$$\mathbf{a}_i^\top \mathbf{x}^* - b_i = 0 \quad \forall i \in \mathcal{E}, \tag{85}$$

$$\mathbf{a}_i^\top \mathbf{x}^* - b_i \geq 0 \quad \forall i \in \mathcal{I}, \tag{86}$$

$$\lambda_i^* \geq 0 \quad \forall i \in \mathcal{I}, \tag{87}$$

$$\lambda_i^*(\mathbf{a}_i^\top \mathbf{x}^* - b_i) = 0 \quad \forall i \in \mathcal{E}\cup\mathcal{I}. \tag{88}$$

The proofs of Theorems 8 and 9 follow the same steps but the the proof of Theorem 9 is slightly shorter. The proof of Theorem 8 is found in Chapter 12 in [1].

**Exercise** Consider a constrained optimization problem with linear equality and inequality constraints:

$$\begin{cases} f(\mathbf{x}) \ \to \ \min \\ A_{\mathcal{E}}\mathbf{x} - b_{\mathcal{E}} = 0 & \text{linear equality contraints} \\ A_{\mathcal{I}}\mathbf{x} - b_{\mathcal{I}} \geq 0 & \text{linear inequality contraints} \end{cases} \tag{89}$$

Reduce (89) to an equivalent problem that has only inequality constraints:

$$\begin{cases} \tilde{f}(\mathbf{y}) \ \to \ \min \\ \tilde{A}\mathbf{y} - \tilde{b} \geq 0 & \text{linear inequality contraints} \end{cases} \tag{90}$$

Write out the expressions for $\tilde{f}$, $\tilde{A}$, and $\tilde{b}$. Suppose you have solved (90), i.e., found a local solution $\mathbf{y}^*$. Write out the formula for the corresponding local solution $\mathbf{x}^*$ of (89).

*Proof.* (Theorem 9.) In the view of the exercise above, it suffices to prove Theorem 9 for the case where the set of constraints consists only of linear inequality constraints, i.e., for the problem

$$f(\mathbf{x}) \to \min \quad \text{subject to} \\ A\mathbf{x}^* - \mathbf{b} \geq 0, \quad A \text{ is } n \times d. \tag{91}$$

Without the loss of generality, we assume that the active set consists of the first $m$ inequalities: $\mathcal{A}(\mathbf{x}^*) = \{1, \ldots, m\}$. The matrix consisting of the first $m$ rows of $A$ and the vector consisting of the first $m$ components of $\mathbf{b}$ will be denoted by $\mathsf{A}$, and $\mathsf{b}$ respectively. Note that there is a neighborhood $\mathcal{U}(\mathbf{x}^*)$ such that $A_{(m+1:n,:)}\mathbf{x} - b_{m+1:n} > 0 \ \forall \mathbf{x} \in \mathcal{U}(\mathbf{x}^*)$.

**Step 1.** *Show that the set of feasible directions coincides with the tangent cone.* Definition 3 implies that the set of feasible directions is

$$\mathcal{F}(\mathbf{x}^*) = \{\mathbf{d} \in \mathbb{R}^d \mid \mathsf{A}\mathbf{d} \geq 0\} \tag{92}$$

and coincides with the tangent cone $T_\Omega(\mathbf{x}^*)$.

- Let $\mathbf{d}$ be a tangent vector. Then there is a feasible sequence $\mathbf{z}_k$ and a sequence of scalars $t_k \to 0$ such that

$$\mathbf{z}_k = \mathbf{x}^* + t_k\mathbf{d} + o(t_k).$$

Multiplying this equation by $\mathsf{A}$ and subtracting $\mathsf{b}$ we get:

$$0 \le \mathsf{A}\mathbf{z}_k - \mathsf{b} = \underbrace{\mathsf{A}\mathbf{x}^* - \mathsf{b}}_{=0} + t_k \mathsf{A}\mathbf{d} + o(t_k) = t_k \mathsf{A}\mathbf{d} + o(t_k).$$

Dividing the last equality by $t_k$ and letting $k \to 0$, we get $\mathsf{A}\mathbf{d} \ge 0$ which means that any tangent vector is a feasible direction, i.e., $T_\Omega(\mathbf{x}^*) \subseteq \mathcal{F}(\mathbf{x}^*)$.

- Let $\mathbf{d}$ be a feasible direction. Show that then there is a sequence of positive scalars $\{t_k\}$ such that the sequence $\mathbf{z}_k = \mathbf{x}^* + t_k\mathbf{d}$ is feasible. We take $\{t_k\}$ sufficiently small so that $\mathbf{z}_k \in \mathcal{U}(\mathbf{x}^*)$. By multiplying this equality by $\mathsf{A}$ and subtracting $\mathsf{b}$ we get

$$\mathsf{A}\mathbf{z}_k - \mathsf{b} = \underbrace{\mathsf{A}\mathbf{x}^* - \mathsf{b}}_{=0} + t_k \mathsf{A}\mathbf{d} = t_k \mathsf{A}\mathbf{d} \ge 0$$

which implies that $\mathbf{d}$ is a tangent vector. Hence $\mathcal{F}(\mathbf{x}^*) \subseteq T_\Omega(\mathbf{x}^*)$.

**Remark** Note that if the constraints are nonlinear and LICQ does not hold, the set of feasible directions does not need to coincide with the tangent cone. For example, the set of feasible directions at $[0,0]^\top$ for problem (82) includes $[0,1]^\top$ and $[0,-1]^\top$, but the tangent cone does not include $[0,-1]^\top$.

**Step 2.** *Show that if* $\mathbf{x}^*$ *is a local solution of* (91) *then*

$$\nabla f(\mathbf{x}^*)^\top \mathbf{d} \ge 0 \quad \forall \mathbf{d} \in T_\Omega(\mathbf{x}^*). \tag{93}$$

Indeed, assume the opposite: there is a tangent vector $\mathbf{d}$ such that $\nabla f(\mathbf{x}^*)^\top \mathbf{d} < 0$. Then there exist sequences $\mathbf{z}_k \to \mathbf{x}^*$, feasible, and $t_k \to 0$, $t_k > 0$, such that

$$f(\mathbf{z}_k) = f(\mathbf{x}^*) + t_k \nabla f(\mathbf{x}^*)^\top \mathbf{d} + o(t_k) \ge f(\mathbf{x}^*).$$

Hence, subtracting $f(\mathbf{x}^*)$ from both sides and dividing by $t_k$, we get:

$$0 \le \frac{f(\mathbf{z}_k) - f(\mathbf{x}^*)}{t_k} = \underbrace{\nabla f(\mathbf{x}^*)^\top \mathbf{d}}_{<0} + o(1) < 0$$

for sufficiently small $t_k$ – a contradiction. Therefore, (93) holds.

**Step 3 (Farkas' lemma).** Consider the cone

$$K := \left\{ \mathsf{A}^\top \lambda \mid \lambda \in \mathbb{R}^m,\ \lambda_i \ge 0\ \forall 1 \le i \le m \right\}. \tag{94}$$

Note that $K$ is convex and closed. Prove that for any vector $\mathbf{g} \in \mathbb{R}^d$ the following alternative takes place:

- **either** $\mathbf{g} \in K$, i.e., there is a vector $\lambda \in \mathbb{R}^m$ with nonnegative components such that $\mathbf{g} = \mathsf{A}^\top \lambda$,
- **or** there exists a vector $\mathbf{d} \in \mathbb{R}^d$ such that the hyperplane normal to $\mathbf{d}$ separates $\mathbf{g}$ and $K$, i.e.,

$$\mathbf{g}^\top \mathbf{d} < 0, \quad \text{while} \quad \mathsf{A}\mathbf{d} \ge 0. \tag{95}$$

First, we show that the two alternatives cannot take place simultaneously. From converse, assume that $\mathbf{g} = \mathsf{A}^\top \lambda$ for $\lambda \geq 0$ while $\mathbf{g}^\top \mathbf{d} < 0$ and $\mathsf{A}\mathbf{d} \geq 0$. Then this leads to a contradiction:

$$0 > \mathbf{g}^\top \mathbf{d} = \boldsymbol{\lambda}^\top \mathsf{A}\mathbf{d} = \underbrace{\lambda^\top}_{\geq 0} \underbrace{\mathsf{A}\mathbf{d}}_{\geq 0} \geq 0.$$

Now we show that if $\mathbf{g} \notin K$ then (95) holds. Let $\mathbf{y}$ be the point of the cone $K$ closest to $\mathbf{g}$ in terms of the Euclidean distance. Such a point exists as $K$ is closed. Note that $\mathbf{y}$ may be the origin or a boundary point of $K$ different from the origin – these two cases are depicted in Fig. 9. In both cases, the argument below is valid.

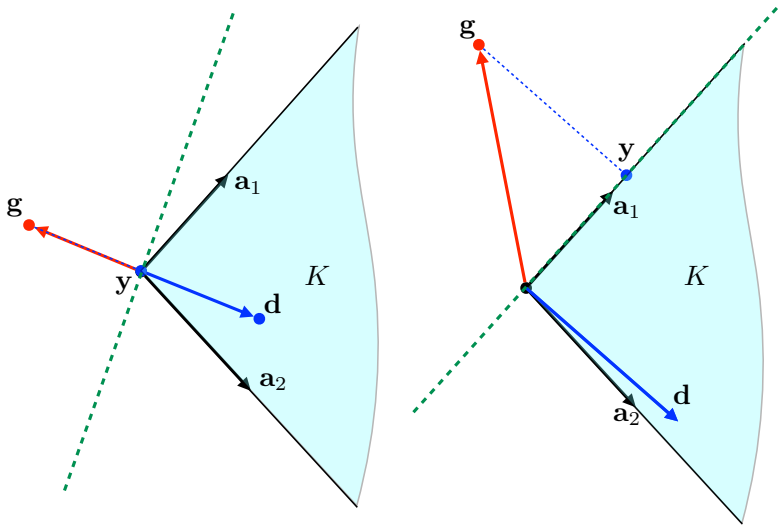


FIGURE 9. Illustration to step 3 of the proof of Theorem 8 in the case where $d = 2$ and $m = 2$. The vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ are the rows of the matrix $\mathsf{A}$.

By the definition of cone, the whole ray emanating from the origin and passing through $\mathbf{y}$ belongs to $K$:

$$\{\alpha \mathbf{y} \in K \mid \alpha \geq 0\}.$$

Since $\mathbf{y}$ is the closest point to $\mathbf{g}$ in this ray as well, the minimum of the function

$$\phi(\alpha) := \frac{1}{2}(\alpha \mathbf{y} - \mathbf{g})^\top (\alpha \mathbf{y} - \mathbf{g}) = \frac{1}{2}\alpha^2 \|\mathbf{y}\|^2 - \alpha \mathbf{y}^\top \mathbf{g} + \frac{1}{2}\|\mathbf{g}\|^2$$

is achieved at $\alpha = 1$. Therefore,

$$0 = \frac{d\phi}{d\alpha}(1) = \|\mathbf{y}\|^2 - \mathbf{y}^\top \mathbf{g} = \mathbf{y}^\top (\mathbf{y} - \mathbf{g}).$$

Let $\mathbf{z} \in K$, $\mathbf{z} \neq \mathbf{y}$. Since $K$ is convex, we have

$$\mathbf{y} + \theta(\mathbf{z} - \mathbf{y}) \in K \quad \forall \theta \in [0, 1].$$

By the minimizing property of $\mathbf{y}$ we have

$$\|\mathbf{y} + \theta(\mathbf{z} - \mathbf{y}) - \mathbf{g}\|^2 \geq \|\mathbf{y} - \mathbf{g}\|^2 \quad \forall \theta \in [0, 1].$$

Hence

$$2\theta(\mathbf{z} - \mathbf{y})^\top(\mathbf{y} - \mathbf{g}) + \theta^2\|\mathbf{z} - \mathbf{y}\|^2 \geq 0.$$

Dividing this equation by $\theta$ and taking limit $\theta \downarrow 0$, we get

$$0 \leq (\mathbf{z} - \mathbf{y})^\top(\mathbf{y} - \mathbf{g}) = \mathbf{z}^\top(\mathbf{y} - \mathbf{g}) - \underbrace{\mathbf{y}^\top(\mathbf{y} - \mathbf{g})}_{=0} = \mathbf{z}^\top(\mathbf{y} - \mathbf{g}), \quad \text{i.e.}$$

$$\mathbf{z}^\top(\mathbf{y} - \mathbf{g}) \geq 0 \quad \forall \mathbf{z} \in K. \tag{96}$$

Now we set

$$\mathbf{d} := \mathbf{y} - \mathbf{g}$$

and show that $\mathbf{d}$ satisfies (95). Note that $\mathbf{d} \neq 0$ as $\mathbf{g} \notin K$. Indeed, (96) means that

$$\mathbf{d}^\top \mathsf{A}^\top \lambda \geq 0 \quad \forall \lambda \geq 0,$$

which is true only if $\mathbf{d}^\top \mathsf{A}^\top \geq 0$, i.e, $\mathsf{A}\mathbf{d} \geq 0$. On the other hand,

$$\mathbf{d}^\top \mathbf{g} = \mathbf{d}^\top(\mathbf{y} - \mathbf{d}) = (\mathbf{y} - \mathbf{g})^\top(\mathbf{y} - \mathbf{d}) = \underbrace{(\mathbf{y} - \mathbf{g})^\top \mathbf{y}}_{=0} - \|\mathbf{d}\|^2 < 0.$$

**Step 4.** Now we set $\mathbf{g} = \nabla f(\mathbf{x}^*)$. By Farkas' lemma proven in Step 3, either $\nabla f(\mathbf{x}^*) \in K$, i.e., there is a vector $\lambda$ with nonnegative components such that

$$\nabla f(\mathbf{x}^*) = \mathsf{A}^\top \lambda, \tag{97}$$

or there is a direction $\mathbf{d}$ such that $\mathsf{A}\mathbf{d} \geq 0$ (i.e. $d$ is feasible) and $\nabla f(\mathbf{x}^*)^\top \mathbf{d} < 0$ which means that $\mathbf{x}^*$ is not a local solution. Therefore, since $\mathbf{x}^*$ is a local solution, (97) takes place.

Finally, we set the Lagrange multipliers with indices in the active set ($1 \leq i \leq m$) to be equal to $\lambda$ and we set to zero the rest of them, i.e.,

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda \\ 0 \end{bmatrix}.$$

The conditions (84)–(88) are readily verified. □

8.2. **Active-set method.** This section is based on J. Nocedal and S. Wright "Numerical Optimization" [1], Chapter 16. We consider a convex quadratic program (QP)

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top G\mathbf{x} + \mathbf{c}^\top \mathbf{x} \ \to \ \min \quad \text{subject to} \tag{98}$$

$$\mathbf{a}_i^\top \mathbf{x} = \mathbf{b}_i, \quad i \in \mathcal{E} \tag{99}$$

$$\mathbf{a}_i^\top \mathbf{x} \geq \mathbf{b}_i, \quad i \in \mathcal{I}. \tag{100}$$

Convexity of the QP means that the matrix $G$ is positive definite.

A quite natural method to solve (98)–(100) is *the active-set method*. At every step of this algorithm, we solve a QP

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top G\mathbf{x} + \mathbf{c}^\top \mathbf{x} \ \to \ \min \quad \text{subject to} \tag{101}$$

$$\mathbf{a}_i^\top \mathbf{x} = \mathbf{b}_i, \quad i \in \mathcal{W}, \tag{102}$$

where $\mathcal{W}$ is the current set of active constraints. We will denote by $A_\mathcal{W}$ the matrix whose rows at $\mathbf{a}_i$, $i \in \mathcal{W}$, and $\mathbf{b}_\mathcal{W}$ the vector of $b_i$'s, $i \in \mathcal{W}$. Then (102) becomes $A_\mathcal{W}\mathbf{x} = \mathbf{b}_\mathcal{W}$.

We do so as follows. Let $\mathbf{x}_k$ be the current iterate. We want to find a step $\mathbf{p}_k$ to obtain the next iterate

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k.$$

In the case if $\mathbf{x}_{k+1}$ is not feasible, we shorten the step length just enough to make $\mathbf{x}_{k+1}$ remain feasible. Plugging $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ into (101) we get

$$\frac{1}{2}(\mathbf{x}_k + \mathbf{p}_k)^\top G(\mathbf{x}_k + \mathbf{p}_k) + \mathbf{c}^\top(\mathbf{x}_k + \mathbf{p}_k) = \frac{1}{2}\mathbf{p}_k^\top G\mathbf{p}_k + (G\mathbf{x}_k + \mathbf{c})^\top \mathbf{p}_k + f(\mathbf{x}_k).$$

We observe that $G\mathbf{x}_k + \mathbf{c} \equiv \nabla f(\mathbf{x}_k)$ and $f(\mathbf{x}_k)$ is independent of $\mathbf{p}_k$ and hence does not affect the minimizer. Moreover, plugging $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ into (102) we get the following constraint for $\mathbf{p}_k$:

$$A_\mathcal{W}(\mathbf{x}_k + \mathbf{p}_k) - \mathbf{b}_\mathcal{W} = \underbrace{A_\mathcal{W}\mathbf{x}_k - \mathbf{b}_\mathcal{W}}_{=0} + A_\mathcal{W}\mathbf{p}_k = A_\mathcal{W}\mathbf{p}_k = 0.$$

Therefore, the minimization problem for $\mathbf{p}_k$ reduces to

$$\frac{1}{2}\mathbf{p}^\top G\mathbf{p} + \nabla f(\mathbf{x}_k)^\top \mathbf{p} \ \to \ \min \quad \text{subject to} \tag{103}$$

$$A_\mathcal{W}\mathbf{p} = 0. \tag{104}$$

The Lagrangian function for (103)–(104) is

$$\mathcal{L}(\mathbf{p}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{p}^\top G\mathbf{p} + \nabla f(\mathbf{x}_k)^\top \mathbf{p} - \boldsymbol{\lambda}^\top A_\mathcal{W}\mathbf{p}. \tag{105}$$

The KKT system for (103)–(104) is

$$G\mathbf{p}_k + \nabla f(\mathbf{x}_k) - A_\mathcal{W}^\top \boldsymbol{\lambda} = 0, \quad A_\mathcal{W}\mathbf{p}_k = 0.$$

It can be rewritten in the matrix form:

$$\begin{bmatrix} G & A_{\mathcal{W}}^\top \\ A_{\mathcal{W}} & 0 \end{bmatrix} \begin{bmatrix} -\mathbf{p}_k \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \nabla f(\mathbf{x}_k) \\ 0 \end{bmatrix}. \tag{106}$$

**Exercise** Show that the matrix in (106) where $G$ is $d \times d$ symmetric positive definite and $A$ is $m \times d$ and has linearly independent rows, is of *saddle-point type*, i.e., it has $d$ positive eigenvalues and $m$ negative ones. *Hint: Omit the subscript $\mathcal{W}$ for brevity. Find matrices $X$ and $S$ ($S$ is called the **Schur compliment**) such that*

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} \begin{bmatrix} G & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & X^\top \\ 0 & I \end{bmatrix}.$$

*Then use Sylvester's Law of Inertia (look it up!) to finish the proof.*

**Exercise** Consider an equality-constrained QP ($G$ is symmetric)

$$\frac{1}{2}\mathbf{x}^\top G\mathbf{x} + \mathbf{c}^\top \mathbf{x} \;\to\; \min \quad \text{subject to} \tag{107}$$

$$A\mathbf{x} = \mathbf{b}. \tag{108}$$

Assume that $A$ is full rank (i.e., its rows are linearly independent) and $Z^\top GZ$ is positive definite where $Z$ is a basis for the null-space of $A$, i.e., $AZ = 0$.

(1) Write the KKT system for this case in the matrix form.
(2) Show that the matrix of this system $K$ is invertible. *Hint: assume that there is a vector $\mathbf{z} := (\mathbf{x}, \mathbf{y})^\top$ such that $K\mathbf{z} = 0$. Consider the form $\mathbf{z}^\top K\mathbf{z}$, and so on ... . You should arrive at the conclusion that then $\mathbf{z} = 0$.*
(3) Conclude that there exists a unique vector $(\mathbf{x}^*, \boldsymbol{\lambda}^*)^\top$ that solves the KKT system. Note that since we have only equality constraints, positivity of $\boldsymbol{\lambda}$ is irrelevant.

According to the claims in the exercises, (106) has a unique solution $(\mathbf{p}, \boldsymbol{\lambda})$. We consider two cases.

- If $\mathbf{p} \neq 0$, we can make a step in the direction $\mathbf{p}$. Let us show that this is a descent direction for $f(\mathbf{x})$, i.e., $\nabla f(\mathbf{x}_k)^\top \mathbf{p} < 0$. Indeed, we have

$$G\mathbf{p} + \nabla f(\mathbf{x}_k) = A_{\mathcal{W}}^\top \boldsymbol{\lambda}$$
$$A_{\mathcal{W}}\mathbf{p} = 0.$$

  Multiplying the first equation by $\mathbf{p}^\top$ we get

$$\mathbf{p}^\top (G\mathbf{p} + \nabla f(\mathbf{x}_k)) = \mathbf{p}^\top G\mathbf{p} + \nabla f(\mathbf{x}_k)^\top \mathbf{p} = \mathbf{p}^\top A_{\mathcal{W}}^\top \boldsymbol{\lambda} = (\underbrace{A_{\mathcal{W}}\mathbf{p}}_{=0})^\top \boldsymbol{\lambda} = 0.$$

  Since $G$ is positive definite and $\mathbf{p} \neq 0$ by assumption, we conclude that $\nabla f(\mathbf{x}_k)^\top \mathbf{p} < 0$ which means that $\mathbf{p}$ is a descent direction for $f$.

  Next, we start moving from $\mathbf{x}_k$ along the direction $\mathbf{p}$ until we either travel the full distance $\|\mathbf{p}\|$ or activate another constraint:

$$\mathbf{a}_i^\top \mathbf{x} = b_i \quad \text{for some} \quad i \notin \mathcal{W}.$$

Hence, to find step length $\alpha$, we consider

$$\mathbf{a}_i^\top(\mathbf{x}_k + \alpha\mathbf{p}) = \mathbf{a}_i^\top\mathbf{x}_k + \alpha\mathbf{a}_i^\top\mathbf{p} = b_i \quad \text{for all} \quad i \notin \mathcal{W}.$$

Since $\mathbf{a}_i^\top\mathbf{x}_k > b_i \ \forall i \notin \mathcal{W}$, if $\mathbf{a}_i^\top\mathbf{p} \geq 0$, we can only reinforce this constraint by moving along $\mathbf{p}$. In contrast, if $\mathbf{a}_i^\top\mathbf{p} < 0$, we may hit the constraint before traveling the full distance. Hence, the step length $\alpha$ is given by

$$\alpha = \min\left\{1, \min_{i\notin\mathcal{W}, \mathbf{a}_i^\top\mathbf{p}<0} \frac{b_i - \mathbf{a}_i^\top\mathbf{x}_k}{\mathbf{a}_i^\top\mathbf{p}}\right\}. \tag{109}$$

Finally, we set $\alpha_k = \alpha$ and $\mathbf{p}_k = \mathbf{p}$.

- If $\mathbf{p} = 0$, we cannot move anywhere from $\mathbf{x}_k$ given the set of constraints $\mathcal{W}$. Hence, there are two possibilities.
  - We have found the solution to (98)–(100). To check if this is the case, we look at the vector $\boldsymbol{\lambda}$ of Lagrange's multipliers, and check their signs. If all Lagrange multipliers corresponding to inequality constraints are nonnegative, a solution is found, and we terminate the iterations.
  - If there is a negative Lagrange multiplier corresponding to an inequality constraint, we remove it from $\mathcal{W}$ and proceed with iterations.

A Matlab function `ASM` implements the active-set method for the case where $f$ is allowed to be nonconvex and nonquadratic, and the set of constraints consists of inequality constraints only. A driver for it with the Rosenbrock function and the feasible region being a hexagon is the function `ASMdriver`.

```
 function ASMdriver()
%%  the Rosenbrock function
a = 5;
func = @(x,y)(1-x).^2 + a*(y - x.^2).^2;  % Rosenbrock's function
gfun = @(x)[-2*(1-x(1))-4*a*(x(2)-x(1)^2)*x(1);2*a*(x(2)-x(1)^2)]; % gradient of f
Hfun = @(x)[2 + 12*a*x(1)^2 - 4*a*x(2), -4*a*x(1); -4*a*x(1), 2*a]; % Hessian of f
lsets = exp([-3:0.5:2]);
%% constraints
Nv = 6;
t = linspace(0,2*pi,Nv+1);
t(end) = [];
t0 = 0.1;
verts = [0.1+cos(t0+t);0.1+sin(t0+t)];
R = [0,-1;1,0];
A = (R*(circshift(verts,[0,-1])-verts))';
b = verts(1,:)'.*A(:,1) + verts(2,:)'.*A(:,2); % b_i = a_i*verts(:,i)
x = [-0.5;0.5];
W = [];
[xiter,lm] = ASM(x,gfun,Hfun,A,b,W);
%% graphics
```

```
close all
fsz = 16;
figure(1);
hold on;
n = 100;
txmin = min(verts(1,:))-0.2;
txmax = max(verts(1,:))+0.2;
tymin = min(verts(2,:))-0.2;
tymax = max(verts(2,:))+0.2;
tx = linspace(txmin,txmax,n);
ty = linspace(tymin,tymax,n);
[txx,tyy] = meshgrid(tx,ty);
ff = func(txx,tyy);
contour(tx,ty,ff,lsets,'Linewidth',1);
edges = [verts,verts(:,1)];
plot(edges(1,:),edges(2,:),'Linewidth',2,'color','k');
plot(xiter(1,:),xiter(2,:),'Marker','.','Markersize',20,'Linestyle','-',...
    'Linewidth',2,'color','r');
xlabel('x','Fontsize',fsz);
ylabel('y','Fontsize',fsz);
set(gca,'Fontsize',fsz);
colorbar;
grid;
daspect([1,1,1]);
end

function [xiter,lm] = ASM(x,gfun,Hfun,A,b,W)
%% minimization using the active set method (Nocedal & Wright, Section 16.5)
% Solves f(x) --> min subject to Ax >= b
% x = initial guess, a column vector
TOL = 1e-10;
dim = length(x);
g = gfun(x);
H = Hfun(x);
iter = 0;
itermax = 1000;
m = size(A,1); % the number of constraints
% W = working set, the set of active constrains
I = (1:m)';
Wc = I; % the compliment of W
xiter = x;
while iter < itermax
    % compute step p: solve 0.5*p'*H*p + g'*p --> min subject to A(W,:)*p = 0
```

```
AW = A(W,:); % LHS of active constraints
% fix H if it is not positive definite
ee = sort(eig(H),'ascend');
if ee(1) < 1e-10
    lam = -ee(1) + 1;
else
    lam = 0;
end
H = H + lam*eye(dim);
if ~isempty(W)
    M = [H, -AW';AW,zeros(size(W,1))];
    RHS = [-g;zeros(size(W,1),1)];
else
    M = H;
    RHS = -g;
end
aux = M\RHS;
p = aux(1:dim);
lm = aux(dim+1:end);
if  norm(p) < TOL  % if step == 0
    if ~isempty(W)
        lm = AW'\g; % find Lagrange multipliers
        if min(lm) >= 0 % if Lagrange multipliers are positive, we are done
            % the minimizer is one of the corners
            fprintf('A local solution is found, iter = %d\n',iter);
            fprintf('x = [\n'); fprintf('%d\n',x);fprintf(']\n');
            break;
        else % remove the index of the most negative multiplier from W
            [lmin,imin] = min(lm);
            W = setdiff(W,W(imin));
            Wc = setdiff(I,W);
        end
    else
        fprintf('A local solution is found, iter = %d\n',iter);
        fprintf('x = [\n'); fprintf('%d\n',x);fprintf(']\n');
        break;
    end
else % if step is nonzero
    alp = 1;
    % check for blocking constraints
    Ap = A(Wc,:)*p;
    icand = find(Ap < -TOL);
    if ~isempty(icand)
```

```
            % find step lengths to all possible blocking constraints
            al = (b(Wc(icand)) - A(Wc(icand),:)*x)./Ap(icand);
            % find minimal step length that does not exceed 1
            [almin,kmin] = min(al);
            alp = min(1,almin);
        end
        x = x + alp*p;
        g = gfun(x);
        H = Hfun(x);
        if alp < 1
            W = [W;Wc(icand(kmin))];
            Wc = setdiff(I,W);
        end
    end
    iter = iter + 1;
    xiter = [xiter,x];
end
if iter == itermax
    fprintf('Stopped because the max number of iterations %d is performed\n',iter);
end
end
```

## References


[1] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2 ed., 2006.

[2] J. W. Demmel, *Applied Numerical Linear Algebra*. SIAM, 1997.

[3] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[4] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 1951.

[5] A. Botev, G. Lever, and D. Barber, "Nesterov's accelerated gradient and momentum as approximations to regularised update descent." arXiv:1607.01981.