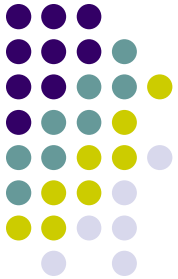
Optimization problems



- General continuous optimization problem:

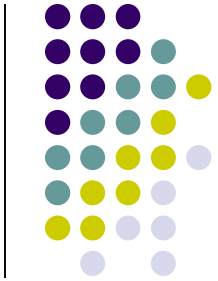
$$\min f(x) \quad \text{subject to} \quad g(x) = 0 \quad \text{and} \quad h(x) \leq 0$$

  where $f: \mathbb{R}^n \to \mathbb{R}$, $g: \mathbb{R}^n \to \mathbb{R}^m$, $h: \mathbb{R}^n \to \mathbb{R}^p$

- *Linear programming*: f, g, and h are all linear

- *Nonlinear programming*: at least one of f, g, and h is nonlinear

Examples



- Minimize weight of structure subject to constraint on its strength, or maximize its strength subject to constraint on its weight
- Minimize cost of diet subject to nutritional constraints
- Minimize surface area of cylinder subject to constraint on its volume:

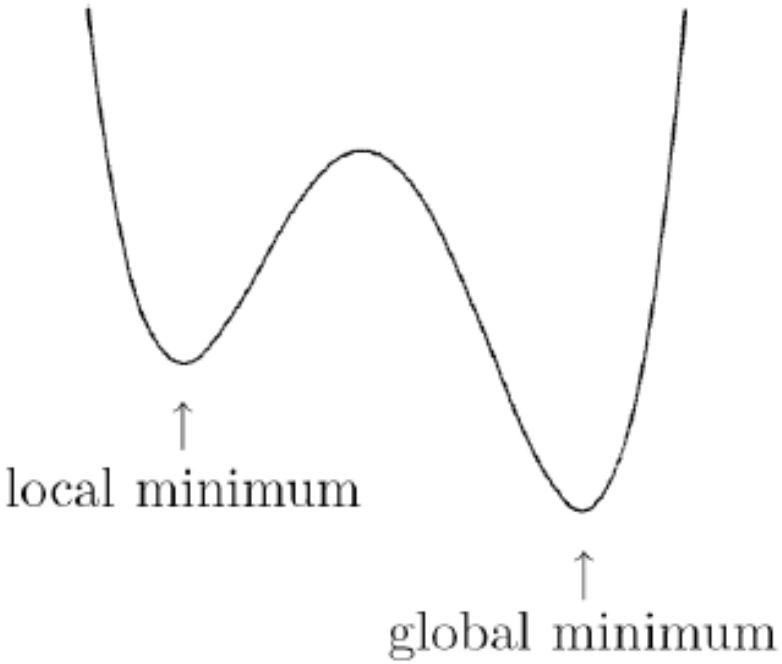
$$\min_{x_1,x_2} f(x_1,x_2) = 2\pi x_1(x_1+x_2)$$

$$\text{subject to} \quad g(x_1,x_2) = \pi x_1^2 x_2 - V = 0$$

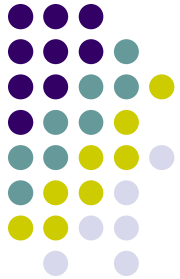
where x_1 and x_2 are radius and height of cylinder, and V is required volume

Global vs. local optimization

- $x^* \in S$ is *global minimum* if $f(x^*) \leq f(x)$ for all $x \in S$
- $x^* \in S$ is *local minimum* if $f(x^*) \leq f(x)$ for all feasible x in some neighborhood of x^*

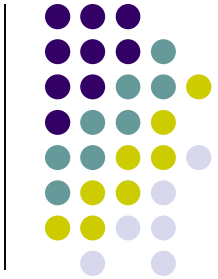


Global optimization



- Finding, or even verifying, global minimum is difficult, in general
- Most optimization methods are designed to find local minimum, which may or may not be global minimum
- If global minimum is desired, one can try several widely separated starting points and see if all produce same result
- For some problems, such as linear programming, global optimization is more tractable

Existence of Minimum



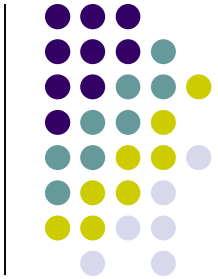
- If f is continuous on *closed* and *bounded* set $S \subseteq \mathbb{R}^n$, then f has global minimum on S
- If S is not closed or is unbounded, then f may have no local or global minimum on S
- Continuous function f on unbounded set $S \subseteq \mathbb{R}^n$ is *coercive* if

$$\lim_{\|\boldsymbol{x}\| \to \infty} f(\boldsymbol{x}) = +\infty$$

  i.e., $f(\boldsymbol{x})$ must be large whenever $\|\boldsymbol{x}\|$ is large

- If f is coercive on closed, unbounded set $S \subseteq \mathbb{R}^n$, then f has global minimum on S

First-order optimality condition



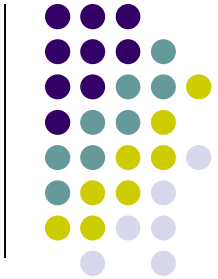
- For function of one variable, one can find extremum by differentiating function and setting derivative to zero
- Generalization to function of n variables is to find *critical point*, i.e., solution of nonlinear system

$$\nabla f(\boldsymbol{x}) = \mathbf{0}$$

  where $\nabla f(\boldsymbol{x})$ is *gradient* vector of f, whose ith component is $\partial f(\boldsymbol{x})/\partial x_i$

- For continuously differentiable $f\colon S \subseteq \mathbb{R}^n \to \mathbb{R}$, any interior point $\boldsymbol{x}^*$ of S at which f has local minimum must be critical point of f
- But not all critical points are minima: they can also be maxima or saddle points

Second-order optimality condition



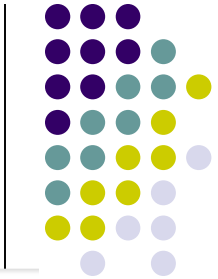
- For twice continuously differentiable $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$, we can distinguish among critical points by considering *Hessian matrix* $\boldsymbol{H}_f(x)$ defined by

$$\{\boldsymbol{H}_f(x)\}_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$$

  which is symmetric

- At critical point x^*, if $\boldsymbol{H}_f(x^*)$ is
  - positive definite, then x^* is minimum of f
  - negative definite, then x^* is maximum of f
  - indefinite, then x^* is saddle point of f
  - singular, then various pathological situations are possible

Constrained optimality



- If problem is constrained, only *feasible* directions are relevant
- For equality-constrained problem

$$\min f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, necessary condition for feasible point $\boldsymbol{x}^*$ to be solution is that negative gradient of f lie in space spanned by constraint normals,

$$-\nabla f(\boldsymbol{x}^*) = \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}^*)\boldsymbol{\lambda}$$

where $\boldsymbol{J}_{\boldsymbol{g}}$ is Jacobian matrix of $\boldsymbol{g}$, and $\boldsymbol{\lambda}$ is vector of *Lagrange multipliers*

- This condition says we cannot reduce objective function without violating constraints

Constrained optimality

- *Lagrangian function* $\mathcal{L}\colon \mathbb{R}^{n+m} \to \mathbb{R}$, is defined by

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$$

- Its gradient is given by

$$\nabla \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_g^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix}$$

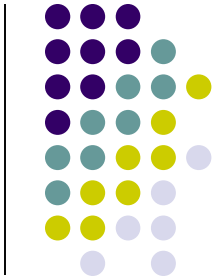
- Its Hessian is given by

$$\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) & \boldsymbol{J}_g^T(\boldsymbol{x}) \\ \boldsymbol{J}_g(\boldsymbol{x}) & \boldsymbol{O} \end{bmatrix}$$

where

$$\boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{H}_f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \boldsymbol{H}_{g_i}(\boldsymbol{x})$$

Constrained optimality

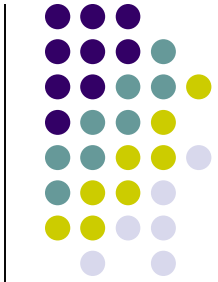


- Together, necessary condition and feasibility imply critical point of Lagrangian function,

$$\nabla \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_g^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix} = \mathbf{0}$$

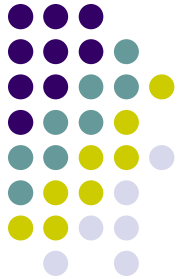
- Hessian of Lagrangian is symmetric, but not positive definite, so critical point of $\mathcal{L}$ is saddle point rather than minimum or maximum
- Critical point (x^*, λ^*) of $\mathcal{L}$ is constrained minimum of f if $B(x^*, \lambda^*)$ is positive definite on *null space* of $J_g(x^*)$
- If columns of Z form basis for null space, then test *projected* Hessian $Z^T B Z$ for positive definiteness

Constrained optimality



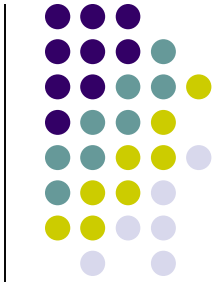
- If inequalities are present, then KKT optimality conditions also require nonnegativity of Lagrange multipliers corresponding to inequalities, and complementarity condition

Unimodality



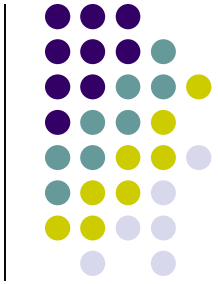
- For minimizing function of one variable, we need "bracket" for solution analogous to sign change for nonlinear equation
- Real-valued function f is *unimodal* on interval $[a, b]$ if there is unique $x^* \in [a, b]$ such that $f(x^*)$ is minimum of f on $[a, b]$, and f is strictly decreasing for $x \leq x^*$, strictly increasing for $x^* \leq x$
- Unimodality enables discarding portions of interval based on sample function values, analogous to interval bisection

Golden section search



- Suppose f is unimodal on $[a, b]$, and let x_1 and x_2 be two points within $[a, b]$, with $x_1 < x_2$
- Evaluating and comparing $f(x_1)$ and $f(x_2)$, we can discard either $(x_2, b]$ or $[a, x_1)$, with minimum known to lie in remaining subinterval
- To repeat process, we need compute only one new function evaluation
- To reduce length of interval by fixed fraction at each iteration, each new pair of points must have same relationship with respect to new interval that previous pair had with respect to previous interval

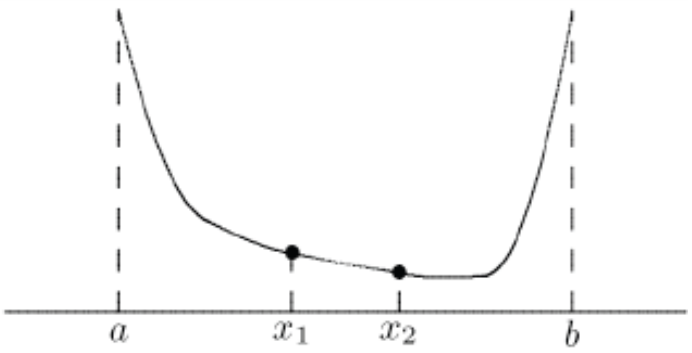
Golden section search

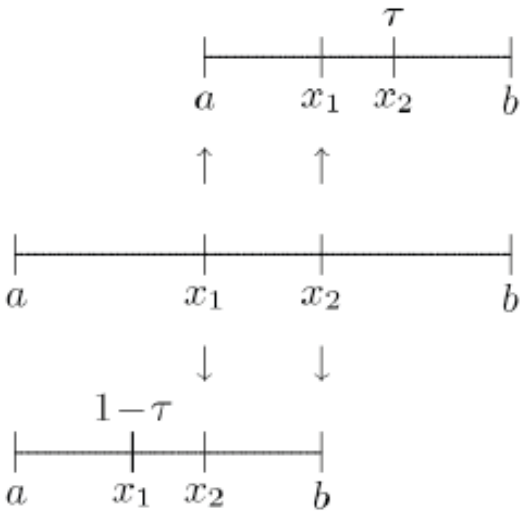


- To accomplish this, we choose relative positions of two points as τ and $1-\tau$, where $\tau^2 = 1-\tau$, so $\tau = (\sqrt{5}-1)/2 \approx 0.618$ and $1-\tau \approx 0.382$
- Whichever subinterval is retained, its length will be τ relative to previous interval, and interior point retained will be at position either τ or $1-\tau$ relative to new interval
- To continue iteration, we need to compute only one new function value, at complementary point
- This choice of sample points is called *golden section search*
- Golden section search is safe but convergence rate is only linear, with constant $C \approx 0.618$

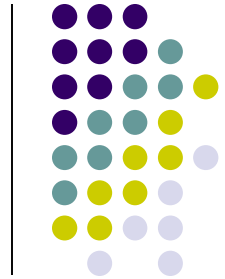
Golden section search

$\tau = (\sqrt{5} - 1)/2$
$x_1 = a + (1 - \tau)(b - a);\ f_1 = f(x_1)$
$x_2 = a + \tau(b - a);\ f_2 = f(x_2)$
while $((b - a) > tol)$ **do**
  if $(f_1 > f_2)$ **then**
    $a = x_1$
    $x_1 = x_2$
    $f_1 = f_2$
    $x_2 = a + \tau(b - a)$
    $f_2 = f(x_2)$
  else
    $b = x_2$
    $x_2 = x_1$
    $f_2 = f_1$
    $x_1 = a + (1 - \tau)(b - a)$
    $f_1 = f(x_1)$
  end
end



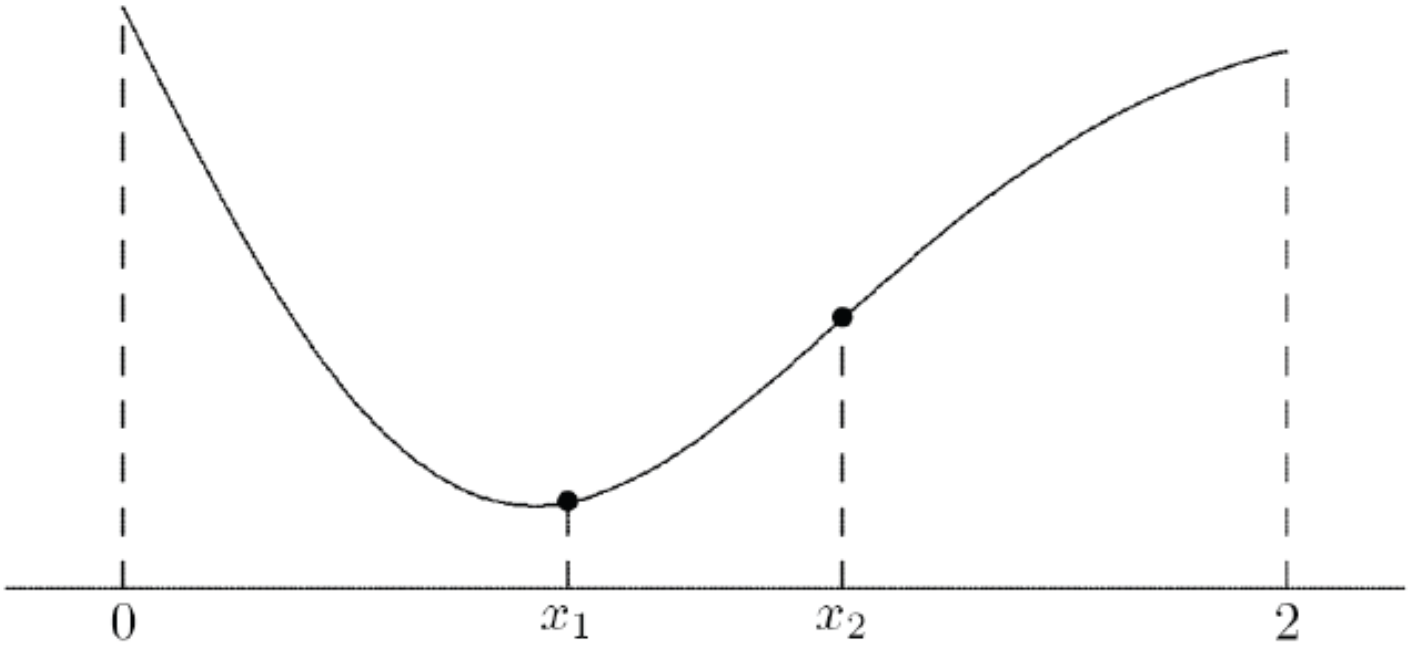


Example

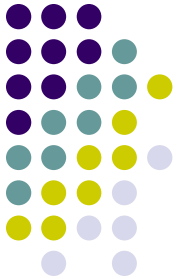


Use golden section search to minimize

$$f(x) = 0.5 - x\exp(-x^2)$$

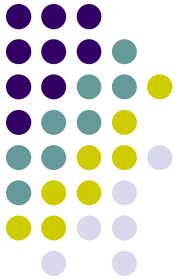


Example (cont.)



x_1	f_1	x_2	f_2
0.764	0.074	1.236	0.232
0.472	0.122	0.764	0.074
0.764	0.074	0.944	0.113
0.652	0.074	0.764	0.074
0.584	0.085	0.652	0.074
0.652	0.074	0.695	0.071
0.695	0.071	0.721	0.071
0.679	0.072	0.695	0.071
0.695	0.071	0.705	0.071
0.705	0.071	0.711	0.071

Newton's method



- Another local quadratic approximation is truncated Taylor series

$$f(x+h) \approx f(x) + f'(x)h + \frac{f''(x)}{2}h^2$$

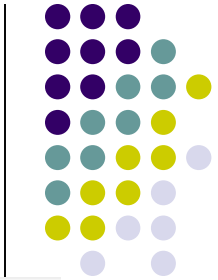
- By differentiation, minimum of this quadratic function of h is given by $h = -f'(x)/f''(x)$
- Suggests iteration scheme

$$x_{k+1} = x_k - f'(x_k)/f''(x_k)$$

which is *Newton's method* for solving nonlinear equation $f'(x) = 0$

Newton's method for finding minimum normally has quadratic convergence rate, but must be started close enough to solution to converge

Example



- Use Newton's method to minimize $f(x) = 0.5 - x\exp(-x^2)$
- First and second derivatives of f are given by

$$f'(x) = (2x^2 - 1)\exp(-x^2)$$

and

$$f''(x) = 2x(3 - 2x^2)\exp(-x^2)$$

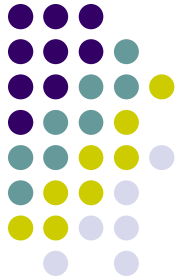
- Newton iteration for zero of f' is given by

$$x_{k+1} = x_k - (2x_k^2 - 1)/(2x_k(3 - 2x_k^2))$$

- Using starting guess $x_0 = 1$, we obtain

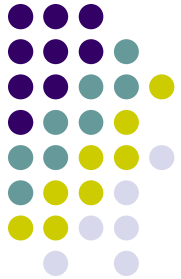
x_k	$f(x_k)$
1.000	0.132
0.500	0.111
0.700	0.071
0.707	0.071

Safeguarded methods



- As with nonlinear equations in one dimension, slow-but-sure and fast-but-risky optimization methods can be combined to provide both safety and efficiency
- Most library routines for one-dimensional optimization are based on this hybrid approach
- Popular combination is golden section search and successive parabolic interpolation, for which no derivatives are required

Multidimensional optimization. Direct search methods



- Direct search methods for multidimensional optimization make no use of function values other than comparing them
- For minimizing function f of n variables, *Nelder-Mead* method begins with $n + 1$ starting points, forming *simplex* in $\mathbb{R}^n$
- Then move to new point along straight line from current point having highest function value through centroid of other points
- New point replaces worst point, and process is repeated
- Direct search methods are useful for nonsmooth functions or for small n, but expensive for larger n

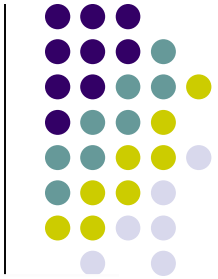
Steepest descent method

- Let $f: \mathbb{R}^n \to \mathbb{R}$ be real-valued function of n real variables
- At any point x where gradient vector is nonzero, negative gradient, $-\nabla f(x)$, points downhill toward lower values of f
- In fact, $-\nabla f(x)$ is locally direction of steepest descent: f decreases more rapidly along direction of negative gradient than along any other
- *Steepest descent* method: starting from initial guess x_0, successive approximate solutions given by

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

where α_k is *line search* parameter that determines how far to go in given direction

Steepest descent method



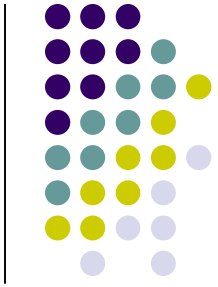
- Given descent direction, such as negative gradient, determining appropriate value for α_k at each iteration is one-dimensional minimization problem

$$\min_{\alpha_k} f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k))$$

  that can be solved by methods already discussed

- Steepest descent method is very reliable: it can always make progress provided gradient is nonzero
- But method is myopic in its view of function's behavior, and resulting iterates can zigzag back and forth, making very slow progress toward solution
- In general, convergence rate of steepest descent is only linear, with constant factor that can be arbitrarily close to 1

Example



- Use steepest descent method to minimize

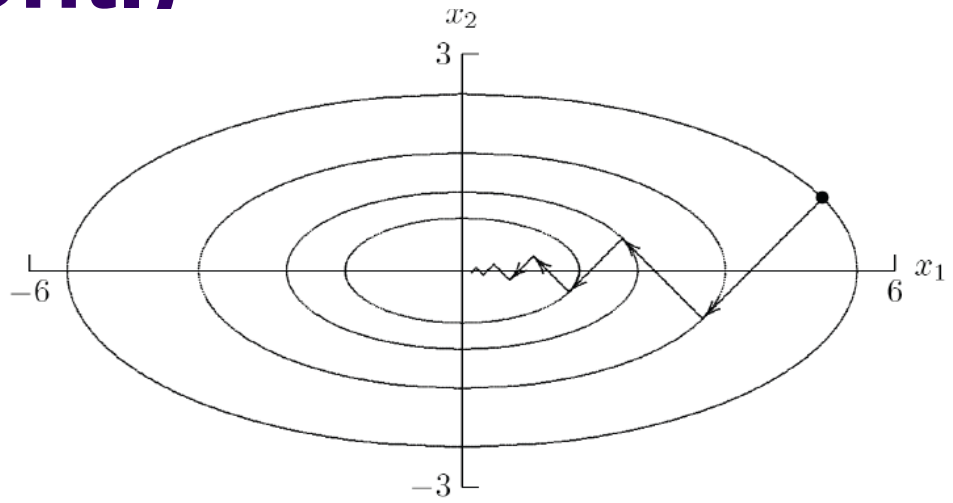
$$f(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2$$

- Gradient is given by $\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix}$
- Taking $\boldsymbol{x}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$, we have $\nabla f(\boldsymbol{x}_0) = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$
- Performing line search along negative gradient direction,

$$\min_{\alpha_0} f(\boldsymbol{x}_0 - \alpha_0 \nabla f(\boldsymbol{x}_0))$$

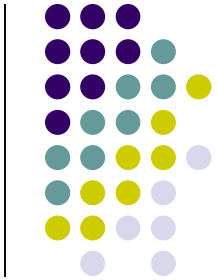
exact minimum along line is given by $\alpha_0 = 1/3$, so next approximation is $\boldsymbol{x}_1 = \begin{bmatrix} 3.333 \\ -0.667 \end{bmatrix}$

Example (cont.)



x_k		$f(x_k)$	$\nabla f(x_k)$	
5.000	1.000	15.000	5.000	5.000
3.333	−0.667	6.667	3.333	−3.333
2.222	0.444	2.963	2.222	2.222
1.481	−0.296	1.317	1.481	−1.481
0.988	0.198	0.585	0.988	0.988
0.658	−0.132	0.260	0.658	−0.658
0.439	0.088	0.116	0.439	0.439
0.293	−0.059	0.051	0.293	−0.293
0.195	0.039	0.023	0.195	0.195
0.130	−0.026	0.010	0.130	−0.130

Newton's method



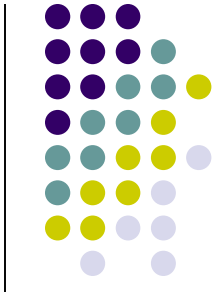
- Broader view can be obtained by local quadratic approximation, which is equivalent to Newton's method
- In multidimensional optimization, we seek zero of gradient, so *Newton iteration* has form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{H}_f^{-1}(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k)$$

where $\boldsymbol{H}_f(\boldsymbol{x})$ is *Hessian* matrix of second partial derivatives of f,

$$\{\boldsymbol{H}_f(\boldsymbol{x})\}_{ij} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}$$

Newton's method



- Do not explicitly invert Hessian matrix, but instead solve linear system

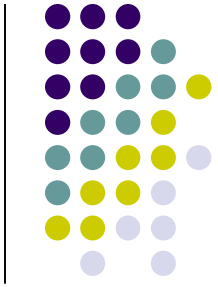
$$\boldsymbol{H}_f(\boldsymbol{x}_k)\boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k)$$

  for Newton step s_k, then take as next iterate

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$$

- Convergence rate of Newton's method for minimization is normally quadratic
- As usual, Newton's method is unreliable unless started close enough to solution to converge

Example



- Use Newton's method to minimize

$$f(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2$$

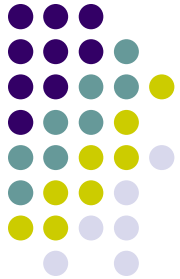
- Gradient and Hessian are given by

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{H}_f(\boldsymbol{x}) = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$$

- Taking $\boldsymbol{x}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$, we have $\nabla f(\boldsymbol{x}_0) = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$
- Linear system for Newton step is $\begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} \boldsymbol{s}_0 = \begin{bmatrix} -5 \\ -5 \end{bmatrix}$, so

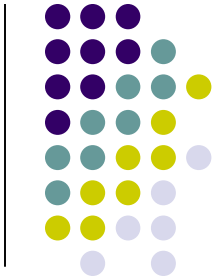
$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{s}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} -5 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, which is exact solution for this problem, as expected for quadratic function

Newton's method



- In principle, line search parameter is unnecessary with Newton's method, since quadratic model determines length, as well as direction, of step to next approximate solution
- When started far from solution, however, it may still be advisable to perform line search along direction of Newton step s_k to make method more robust (*damped* Newton)
- Once iterates are near solution, then $\alpha_k = 1$ should suffice for subsequent iterations

Newton's method

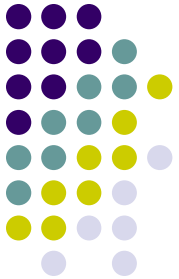


- If objective function f has continuous second partial derivatives, then Hessian matrix $\boldsymbol{H}_f$ is symmetric, and near minimum it is positive definite
- Thus, linear system for step to next iterate can be solved in only about half of work required for LU factorization
- Far from minimum, $\boldsymbol{H}_f(\boldsymbol{x}_k)$ may not be positive definite, so Newton step $\boldsymbol{s}_k$ may not be *descent direction* for function, i.e., we may not have

$$\nabla f(\boldsymbol{x}_k)^T \boldsymbol{s}_k < 0$$

- In this case, alternative descent direction can be computed, such as negative gradient or direction of negative curvature, and then perform line search

Quasi-Newton methods



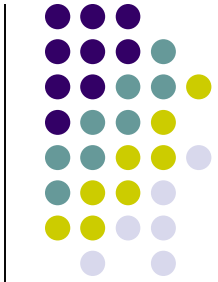
- Newton's method costs $\mathcal{O}(n^3)$ arithmetic and $\mathcal{O}(n^2)$ scalar function evaluations per iteration for dense problem
- Many variants of Newton's method improve reliability and reduce overhead
- *Quasi-Newton* methods have form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{B}_k^{-1} \nabla f(\boldsymbol{x}_k)$$

  where α_k is line search parameter and B_k is approximation to Hessian matrix

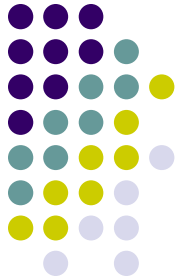
- Many quasi-Newton methods are more robust than Newton's method, are superlinearly convergent, and have lower overhead per iteration, which often more than offsets their slower convergence rate

Conjugate gradient method



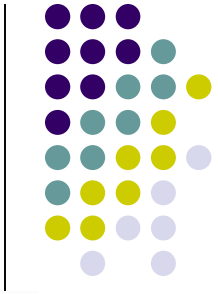
- Another method that does not require explicit second derivatives, and does not even store approximation to Hessian matrix, is *conjugate gradient* (CG) method
- CG generates sequence of conjugate search directions, implicitly accumulating information about Hessian matrix
- For quadratic objective function, CG is theoretically exact after at most n iterations, where n is dimension of problem
- CG is effective for general unconstrained minimization as well

Inequality-constrained optimization



- Methods just outlined for equality constraints can be extended to handle inequality constraints by using *active set* strategy
- Inequality constraints are provisionally divided into those that are satisfied already (and can therefore be temporarily disregarded) and those that are violated (and are therefore temporarily treated as equality constraints)
- This division of constraints is revised as iterations proceed until eventually correct constraints are identified that are binding at solution

Penalty methods



- Merit function can also be used to convert equality-constrained problem into sequence of unconstrained problems
- If x_ρ^* is solution to

$$\min_{\boldsymbol{x}} \phi_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \tfrac{1}{2}\,\rho\,\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{g}(\boldsymbol{x})$$

then under appropriate conditions

$$\lim_{\rho\to\infty} \boldsymbol{x}_\rho^* = \boldsymbol{x}^*$$

This enables use of unconstrained optimization methods, but problem becomes ill-conditioned for large ρ, so we solve sequence of problems with gradually increasing values of , with minimum for each problem used as starting point for next problem