ASTR8150/PHYS8150
Optimization

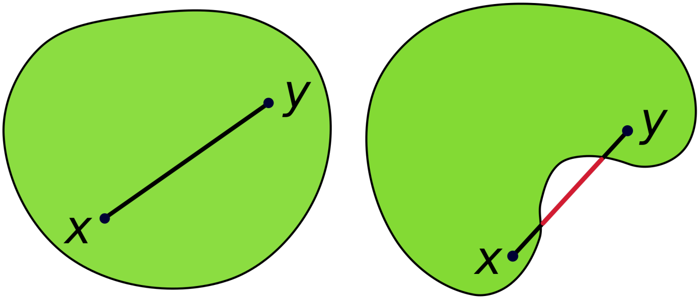
Fabien Baron

Georgia State University

baron@chara.gsu.edu

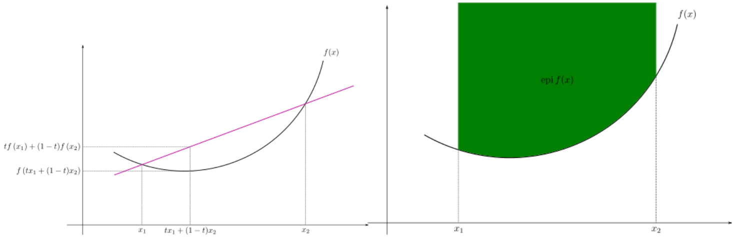
Fall 2019

Convexity of a set



- In a convex set, for every pair of points within the region, every point on the straight line segment that joins the pair of points is also within the region.
- A set which is hollow or has an indent, for example, a crescent shape, is not convex.

Convexity of a function



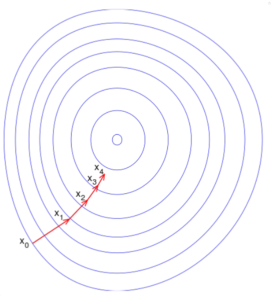
- A real-valued function is called is convex if the set of points on or above the graph of the function (epigraph) is a convex set.
- For a twice differentiable function of a single variable, if the second derivative is always greater than or equal to zero for its entire domain then the function is convex. Examples: $f(x) = x^2$ or $f(x) = e^x$
- Jensen's inequality: if X is a convex set and $f : X \to \mathbb{R}$, f is convex if:

$$\forall x_1, x_2 \in X, \forall t \in [0,1] : \quad f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

Smoothness of a function

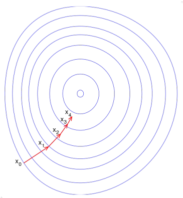
- The smoothness of a function is a property measured by the number of derivatives it has which are continuous. A smooth function is a function that has derivatives of all orders everywhere in its domain.
- The function $f(x) = |x|^k$ is continuous and k times differentiable at all x. But at $x = 0$ they are not $(k+1)$ times differentiable.
- The norms ℓ_2, ℓ_1 and pseudo-norm ℓ_0 are used in regularization. ℓ_2 is convex, differentiable and smooth. ℓ_1 is convex, differentiable but nonsmooth. ℓ_0 is non-convex and nonsmooth.

Gradient descent (1)



- Gradient descent is based on the observation that if the multi-variable function $f(\mathbf{x})$ is defined differentiable in a neighborhood of a point $\mathbf{x_0}$, then $F(\mathbf{x})$ decreases "fastest" if one goes from $\mathbf{x_0}$ in the direction of the negative gradient of f at $\mathbf{x_0}$, $-\nabla f(\mathbf{x_0})$.

Gradient descent (2)



- It follows that, if

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla f(\mathbf{x}_n) \tag{1}$$

for α small enough, then $f(\mathbf{x_n}) \geq f(\mathbf{x_{n+1}})$. In other words, the term $\alpha \nabla f(\mathbf{x})$ is subtracted from $\mathbf{x}$ because we want to move against the gradient, namely down toward the minimum.

- How can we choose α ?
- The Rosenbrock function $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - {x_1}^2)^2$. has a narrow curved valley which contains the minimum. The bottom of the valley is very flat. Because of the curved flat valley the optimization is zig-zagging slowly with small stepsizes towards the minimum.

Steepest descent using line search

- **Inexact line search** consist of finding $\alpha_k \simeq \underset{\alpha \in \mathbb{R}_+}{\operatorname{argmin}} \; f(\mathbf{x}_k + \alpha \mathbf{d}_k)$
- The **line search** method is one of two basic iterative approaches to find a local minimum $\mathbf{x}^*$ of an objective function $f : \mathbb{R}^n \to \mathbb{R}$ using gradients. The other approach is **trust region**.

Algorithm 1 Steepest descent with line search

```
procedure STEEPEST DESCENT($f, \mathbf{x}$)
    $k = 0$, $\mathbf{x}_0$                                ▷ Iteration counter + initial parameter guess
    while $\|\nabla f(\mathbf{x}_k)\| > \epsilon$ do                         ▷ $\epsilon$ = tolerance
        $\mathbf{d_k} = -\nabla f(x_k)$                ▷ Descent direction = Steepest descent
        $\alpha_k \simeq \underset{\alpha \in \mathbb{R}_+}{\operatorname{argmin}} \; f(\mathbf{x}_k + \alpha \mathbf{d}_k)$                          ▷ Line search
        $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
        $k = k + 1$
    end while
end procedure
```

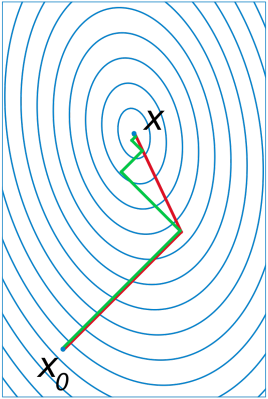
Nonlinear conjugate gradient methods

- Let's pose $\mathbf{g_k} = \nabla f(x_k)$
- Conjugate directions deviate from the steepest descent $\mathbf{d_k} = -\mathbf{g_k}$ by attempting moves based on the history of the previous moves
- The descent direction for nonlinear conjugate gradient methods is

$$\mathbf{d_{k+1}} = -\mathbf{g_{k+1}} + \beta_k \mathbf{d_k}, \quad \mathbf{d_0} = -\mathbf{g_0} \tag{2}$$

- The variation of the gradient is measured by $\mathbf{y_k} = \mathbf{g_{k+1}} - \mathbf{g_k}$
- The Conjugate Gradient update parameter β_k can be updated with different formulas

Conjugate gradient: convergence



- A comparison of the linear convergence of simple gradient descent with optimal step size (in green) and the superlinear convergence of conjugate gradient (in red) for minimizing a quadratic function.

Nonlinear conjugate gradient methods

$\beta_k^{HS} = \dfrac{\mathbf{g}_{k+1}^\mathsf{T}\mathbf{y}_k}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}$	(1952)	in the original (linear) CG paper of Hestenes and Stiefel [59]
$\beta_k^{FR} = \dfrac{\lVert\mathbf{g}_{k+1}\rVert^2}{\lVert\mathbf{g}_k\rVert^2}$	(1964)	first nonlinear CG method, proposed by Fletcher and Reeves [45]
$\beta_k^{D} = \dfrac{\mathbf{g}_{k+1}^\mathsf{T}\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k}{\mathbf{d}_k^\mathsf{T}\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k}$	(1967)	proposed by Daniel [39], requires evaluation of the Hessian $\nabla^2 f(\mathbf{x})$
$\beta_k^{PRP} = \dfrac{\mathbf{g}_{k+1}^\mathsf{T}\mathbf{y}_k}{\lVert\mathbf{g}_k\rVert^2}$	(1969)	proposed by Polak and Ribière [84] and by Polyak [85]
$\beta_k^{CD} = \dfrac{\lVert\mathbf{g}_{k+1}\rVert^2}{-\mathbf{d}_k^\mathsf{T}\mathbf{g}_k}$	(1987)	proposed by Fletcher [44], CD stands for “Conjugate Descent”
$\beta_k^{LS} = \dfrac{\mathbf{g}_{k+1}^\mathsf{T}\mathbf{y}_k}{-\mathbf{d}_k^\mathsf{T}\mathbf{g}_k}$	(1991)	proposed by Liu and Storey [67]
$\beta_k^{DY} = \dfrac{\lVert\mathbf{g}_{k+1}\rVert^2}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}$	(1999)	proposed by Dai and Yuan [27]
$\beta_k^{N} = \left(\mathbf{y}_k - 2\mathbf{d}_k\dfrac{\lVert\mathbf{y}_k\rVert^2}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}\right)^\mathsf{T}\dfrac{\mathbf{g}_{k+1}}{\mathbf{d}_k^\mathsf{T}\mathbf{y}_k}$	(2005)	proposed by Hager and Zhang [53]

Newton optimization method

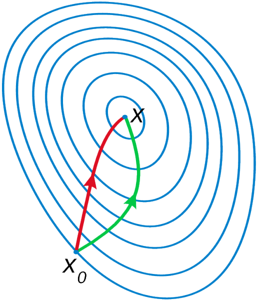


Figure: A comparison of gradient descent (green) and Newton's method (red) for minimizing a function (with small step sizes). Newton's method uses curvature information to take a more direct route.

- Hessian is used to exploit the curvature information

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha[\mathbf{H}f(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n) \tag{3}$$

- $\alpha \in (0, 1)$, with $\alpha = 1$ the exact form.

Newton-Raphson root-finding and eccentric anomaly

- Newton optimization method and Newton-Raphson's root finding methods are based on similar principles
- Newton-Raphson: $x_{n+1} = x_n - f(x_n)/f'(x_n)$
- Example: the mean anomaly is proportional to time it is an easily measured quantity for an orbiting body. Given the mean anomaly M, find the eccentric anomaly E and the orbital eccentricity e with Kepler's Equation:

$$M = E - e \sin E \tag{4}$$

Better than Newton: quasi-Newton methods

- Also known as variable metric methods, they avoid computing the Hessian then its inverse.
- The Broyden-Fletcher-Goldfarb-Shanno (BFGS) or Davidon-Fletcher-Powell (DFP) algorithms build iteratively approximations of $[\mathbf{H}f(\mathbf{x}_n)]^{-1}$.
- The most successfull is Limited-memory BFGS (L-BFGS) that approximates $[\mathbf{H}f(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n)$ directly and thus can work on large scale problems (millions of variables).
- There are variants that attempt to deal with non-smooth functions (subgradient and bundle method)
- There are variants that deal with constrained miminization (i.e. bounds on variables or linearly tied variables) such as L-BFGS-B. Further refinements led to the VMLM algorithm in OptimPack.

Trust-region method and Levenberg-Marquardt

- Consider the quadratic approximation of function f around x_0:

$$q(\epsilon) \simeq f(x_0) + \nabla f(x_0)\epsilon + \frac{1}{2}\epsilon^T \nabla^2 f(x_0)\epsilon \tag{5}$$

- $q(\epsilon)$ has a close-form mimimum.
- $q(\epsilon)$ remains a good approximation within a given radius, $||e||_2 < r^2$ defines the **trust region** radius r.
- The quadratic approximation predicts a certain reduction in the cost function, Δf_{pred}, which is compared to the true reduction $\Delta f_{\text{actual}} = f(x) - f(x + \epsilon)$. By looking at the ratio $\Delta f_{\text{pred}}/\Delta f_{\text{actual}}$ we can estimate the trust-region size at each iteration, jump to the closed-form minimum within the trust region, and iterate.
- The Levenberg-Marquardt algorithm (first published in 1944 by Kenneth Levenberg, rediscovered in 1963 by Donald Marquardt) uses the trust-region approach with conjugate-gradients and Gauss-Newton (Newton optimized for non-linear χ^2). Like conjugate gradient and Newton, these local optimization.

Derivative-free optimization methods

- **Derivative-free** optimization methods simply do not require gradient information
- Among the most popular local optimizer is Nelder–Mead method (aka downhill simplex method or amoeba method), which moves points of a polytope of $n + 1$ vertices in n-parameter dimensions via reflection, contraction, expansion steps
- Most MCMC optimization methods.
- NLopt library provides mostly derivative-free algorithms, some of them for global optimization.

Constrained minimization: the Lagrangian method

- **Constrained** minimization is minimization under equality or inequality constrains. **Bounded** optimization is a special case of constrained optimization where bounds are imposed on parameters (e.g. positivity, or variable within a range). In Bayesian terms, we're imposing a prior.
- A classic example is:

$$(\tilde{x}, \tilde{y}) = \underset{(x,y)\in\mathbb{R}^2}{\text{argmin}} \ (x+y) \quad \text{s. t.} \ \ x^2 + y^2 = 1$$

- We pose $g(x, y) = x^2 + y^2 - 1$ and the Lagrangian is:

$$\begin{aligned}\mathcal{L}(x, y, \lambda) &= f(x, y) + \lambda \cdot g(x, y) \\ &= x + y + \lambda(x^2 + y^2 - 1).\end{aligned}$$

where λ is a Lagrange multiplier.

Constrained minimization: the Lagrangian method

- The gradient with respect to variables x,y and λ

$$\nabla_{x,y,\lambda}\mathcal{L}(x,y,\lambda) = \left(\frac{\partial\mathcal{L}}{\partial x}, \frac{\partial\mathcal{L}}{\partial y}, \frac{\partial\mathcal{L}}{\partial\lambda}\right)$$
$$= \left(1+2\lambda x, 1+2\lambda y, x^2+y^2-1\right)$$

and therefore:

$$\nabla_{x,y,\lambda}\mathcal{L}(x,y,\lambda) = 0 \quad \Leftrightarrow \quad \begin{cases} 1+2\lambda x = 0 \\ 1+2\lambda y = 0 \\ x^2+y^2-1 = 0 \end{cases}$$

- Solution $x = y = -\frac{1}{2\lambda}$, $\lambda \neq 0$.. Substituting into the last equation we get $\lambda = \pm\frac{1}{\sqrt{2}}$ which implies that the stationary points of $\mathcal{L}$ are $\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, -\frac{1}{\sqrt{2}}\right), \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}, \frac{1}{\sqrt{2}}\right)$. And since $f\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) = \sqrt{2}$ and $f\left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right) = -\sqrt{2}$. the solution is found.

Constrained minimization: Half-quadratic splitting (1)

- Let's say we want to minimize:

$$\tilde{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \right\|_2^2 + \mu \Phi(\boldsymbol{x})$$

  where $\Phi(x)$ is a function such that we wouldn't be able to solve this via Tikhonov. We saw a good example is $\Phi(\boldsymbol{x}) = \ell_1(\boldsymbol{x})$.

- Splitting methods are methods that split the unconstrained problem into a constrained problem, using two different variables to represent the same one in different functions:

$$\begin{aligned} \tilde{\boldsymbol{x}} &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \right\|_2^2 + \mu \Phi(\boldsymbol{z}) \quad s.t. \quad \boldsymbol{z} = \boldsymbol{x} \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \right\|_2^2 + \mu \Phi(\boldsymbol{z}) + \frac{\rho}{2} \left\| z - x \right\|_2^2 \end{aligned}$$

Constrained minimization: the two subproblems

- So we now want to minimize:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \mu\Phi(\boldsymbol{z}) + \frac{\rho}{2} \|\boldsymbol{z} - \boldsymbol{x}\|_2^2$$

- $\frac{\rho}{2} \|\boldsymbol{z} - \boldsymbol{x}\|_2^2$ is called an augmented term, and ρ the augmented penalty or hyperparameter.
- The **half-quadratic splitting method** solves iteratively (iteration variable $= k$) the problem with respect to $\boldsymbol{x}$, then $\boldsymbol{z}$:

$$\tilde{\boldsymbol{x}}^{k+1} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \frac{\rho}{2} \left\|\tilde{\boldsymbol{z}}^k - \boldsymbol{x}\right\|_2^2 \qquad \boldsymbol{x} \text{ sub-problem}$$

$$\tilde{\boldsymbol{z}}^{k+1} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\rho}{2} \left\|\boldsymbol{z} - \tilde{\boldsymbol{x}}^{k+1}\right\|_2^2 + \mu\Phi(\boldsymbol{z}) \qquad \boldsymbol{z} \text{ sub-problem}$$

and then increases ρ from initially low values to higher and higher ones.

Constrained minimization: analytical solutions exist

- Why is this easier than the original problem ?

$$\tilde{\boldsymbol{x}}^{k+1} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \frac{\rho}{2} \left\| \tilde{\boldsymbol{z}}^k - \boldsymbol{x} \right\|_2^2 \qquad \boldsymbol{x} \text{ sub-problem}$$

$$\tilde{\boldsymbol{z}}^{k+1} = \underset{\boldsymbol{z}}{\operatorname{argmin}} \frac{\rho}{2} \left\| \boldsymbol{z} - \tilde{\boldsymbol{x}}^{k+1} \right\|_2^2 + \mu\Phi(\boldsymbol{z}) \qquad \boldsymbol{z} \text{ sub-problem}$$

- The $\boldsymbol{x}$ sub-problem can be solved by Tikhonov.
- The $\boldsymbol{z}$ sub-problem can be solved analytically for some functions, for which we know the solution of the problem:

$$\operatorname{prox}_f(\boldsymbol{z}) = \underset{\boldsymbol{y}}{\operatorname{argmin}} \left(\frac{1}{2} \|\boldsymbol{z} - \boldsymbol{y}\|_2^2 + f(\boldsymbol{y}) \right)$$

- This solution $\operatorname{prox}_f(\boldsymbol{z})$ is the **proximal operator** for the function f. At each point $\boldsymbol{z}$ it finds a close-by local minimum of f.

Proximal operator for the ℓ_1 norm and positivity

- One can demonstrate that the proximal operator for ℓ_1 norm is:

$$\text{prox}_{\alpha\ell_1}(\boldsymbol{z}) = \underset{\boldsymbol{y}}{\text{argmin}} \left(\frac{1}{2} \|\boldsymbol{z} - \boldsymbol{y}\|_2^2 + \alpha\ell_1(\boldsymbol{y}) \right) = \text{sign}(\boldsymbol{z}) \cdot \max(\boldsymbol{z} - \alpha, 0)$$

  where $\cdot$ is the Hadamard product.

- The proximal operator for positivity is the projection onto the positive set:

$$\text{prox}_{I_{\mathbb{R}+}}(\boldsymbol{z}) = \underset{\boldsymbol{y} \in \mathbb{R}^{+n}}{\text{argmin}} \left(\frac{1}{2} \|\boldsymbol{z} - \boldsymbol{y}\|_2^2 \right) = \max(\boldsymbol{z}, 0)$$

Half-quadratic splitting: beyond the 1:1 change of variable

- Half-quadratic splitting only involves analytical steps: the $\boldsymbol{x}$ sub-problem is solved via Tikhonov and the $\boldsymbol{z}$ sub-problem via proximal operators (provided it is known). Instead of posing $\boldsymbol{z} = \boldsymbol{x}$, we can also have more complex constrains $\boldsymbol{Az} + \boldsymbol{Bx} + \boldsymbol{c} = 0$.
- How should we solve the total variation problem, i.e. minimize $\frac{1}{2}\left\|\boldsymbol{Ax} - \boldsymbol{b}\right\|_2^2 + \mu\ell_1(\boldsymbol{\nabla x})$?
- We only know the proximal operator for $\ell_1(\boldsymbol{z})$ and not for $\ell_1(\boldsymbol{\nabla z})$. So we should not pose $\boldsymbol{z} = \boldsymbol{x}$, since we wouldn't know how to solve the $\boldsymbol{z}$ sub-problem. However we can pose $\boldsymbol{z} = \boldsymbol{\nabla x}$, leading to:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{2}\left\|\boldsymbol{Ax} - \boldsymbol{b}\right\|_2^2 + \mu\ell_1(\boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{z} - \boldsymbol{\nabla x}\right\|_2^2$$

Total variation solved via Half-quadratic splitting

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{z}) = \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 + \mu\ell_1(\boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{z}-\boldsymbol{\nabla}\boldsymbol{x}\right\|_2^2$$

- The x sub-problem is:

$$\begin{aligned}\tilde{\boldsymbol{x}}^{k+1} &= \underset{\boldsymbol{x}}{\operatorname{argmin}}\,\frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\right\|_2^2 + \frac{\rho}{2}\left\|\tilde{\boldsymbol{z}}^k-\boldsymbol{\nabla}\boldsymbol{x}\right\|_2^2 \qquad \boldsymbol{x}\text{ sub-problem}\\ &\implies \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})-\rho\boldsymbol{\nabla}^\top(\tilde{\boldsymbol{z}}^k-\boldsymbol{\nabla}\boldsymbol{x})=0\\ &\implies x = (\boldsymbol{A}^\top\boldsymbol{A}+\rho\boldsymbol{\nabla}^\top\boldsymbol{\nabla})^{-1}(\boldsymbol{A}^\top\boldsymbol{b}+\rho\boldsymbol{\nabla}^\top\tilde{\boldsymbol{z}}^k)\end{aligned}$$

- The z sub-problem is:

$$\begin{aligned}\tilde{\boldsymbol{z}}^{k+1} &= \underset{\boldsymbol{z}}{\operatorname{argmin}}\,\mu\ell_1(\boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{z}-\boldsymbol{\nabla}\tilde{\boldsymbol{x}}^{k+1}\right\|_2^2 \qquad \boldsymbol{z}\text{ sub-problem}\\ &= \operatorname{sign}(\boldsymbol{z_0})\cdot\max(\boldsymbol{z_0}-\alpha,0)\quad\text{with } \boldsymbol{z}_0=\boldsymbol{\nabla}\tilde{\boldsymbol{x}}^{k+1}\text{ and } \alpha=\frac{\mu}{\rho}\end{aligned}$$