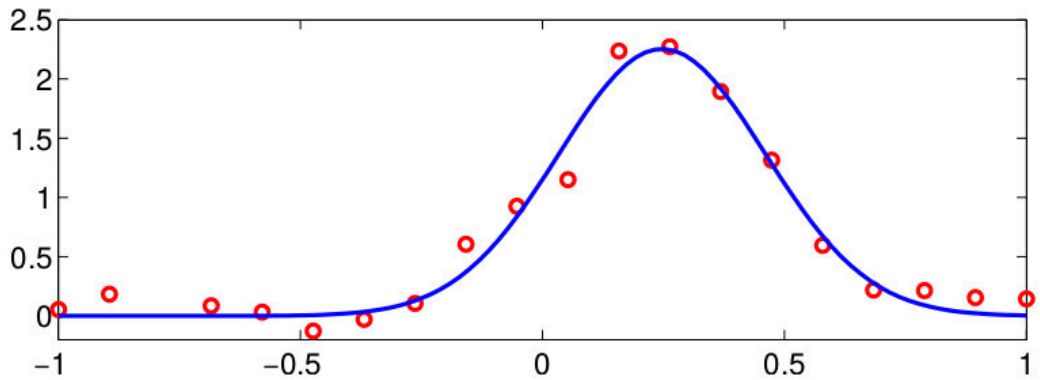
Nonlinear Least Squares

Until now we had linear in parameters functions to fit the data. This is generally not the case for real problems.

Some examples of non-linear fitting functions would be exponential decay function $M(x, c_1, c_2) = c_1 e^{-c_2 x}$ or non-normalized Gaussian function $M(x, \boldsymbol{c}) = c_1 e^{-(x-c_2)^2/c_3^2}$

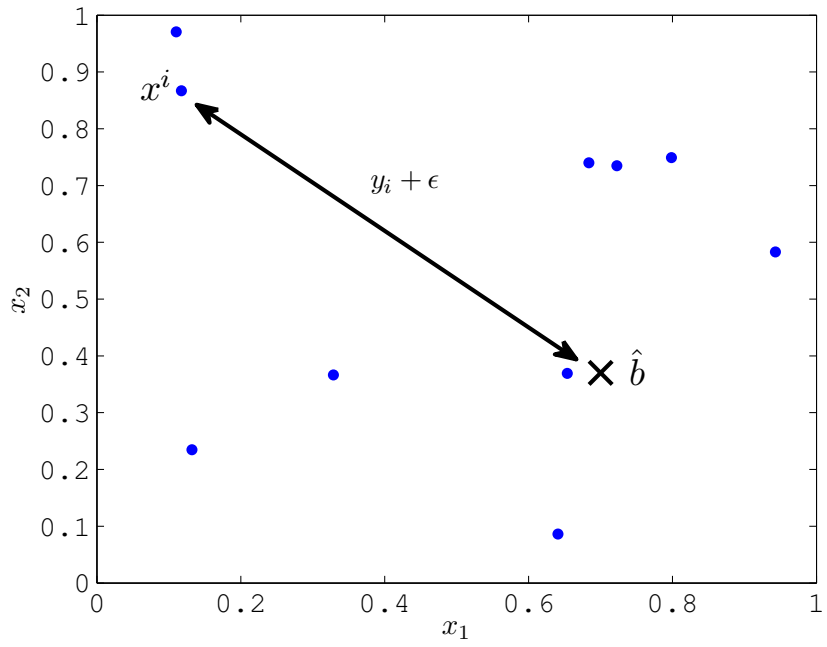


Nonlinear Least Squares: Example

Example: Suppose we have a radio transmitter at $\hat{b} = (\hat{b}_1, \hat{b}_2)$ somewhere in $[0,1]^2$ ($\times$)

Suppose that we have 10 receivers at locations $(x_1^1, x_2^1), (x_1^2, x_2^2), \ldots, (x_1^{10}, x_2^{10}) \in [0,1]^2$ ($\bullet$)

Receiver i returns a measurement for the distance y_i to the transmitter, but there is some error/noise (ϵ)



Nonlinear Least Squares: Example

Let b be a candidate location for the transmitter

The distance from b to (x_1^i, x_2^i) is

$$d_i(b) \equiv \sqrt{(b_1 - x_1^i)^2 + (b_2 - x_2^i)^2}$$

We want to choose b to match the data as well as possible, hence minimize the residual $r(b) \in \mathbb{R}^{10}$ where $r_i(b) = y_i - d_i(b)$

Nonlinear Least Squares: Example

In this case, $r_i(\alpha + \beta) \neq r_i(\alpha) + r_i(\beta)$, hence nonlinear least-squares!

Define the objective function $\phi(b) = \frac{1}{2}\|r(b)\|_2^2$, where $r(b) \in \mathbb{R}^{10}$ is the residual vector

The $1/2$ factor in $\phi(b)$ has no effect on the minimizing b, but leads to slightly cleaner formulae later on

Nonlinear Least Squares

As usually we will try to minimize the objective function by differentiating it and setting to zero:

$$\phi(b) = \frac{1}{2}||r(b)||^2 = \frac{1}{2}\sum_{j=1}^{k}[r_j(b)]^2$$

$$\frac{\partial \phi}{\partial b_i} = \frac{\partial}{\partial b_i}\frac{1}{2}\sum_{j=1}^{k} r_j^2 = \sum_{j=1}^{k} r_j \frac{\partial r_j}{\partial b_i}$$

Nonlinear Least Squares

As usually we will try to minimize the objective function by differentiating it and setting to zero:

$$\phi(b) = \frac{1}{2}||r(b)||^2 = \frac{1}{2}\sum_{j=1}^{k}[r_j(b)]^2$$

$$\frac{\partial \phi}{\partial b_i} = \frac{\partial}{\partial b_i}\frac{1}{2}\sum_{j=1}^{k} r_j^2 = \sum_{j=1}^{k} r_j \frac{\partial r_j}{\partial b_i}$$

Denoting $J_r(b) = \{\frac{\partial r_j}{\partial b_i}\}_{ij}$ the Jacobian matrix we have

$$\nabla \phi = J_r(b)^T r(b)$$

Nonlinear Least Squares

To find the minimum of objective function we need to solve the equation

$$J_r(b)^T r(b) = 0$$

This system has n equations and n unknowns, but most likely is a nonlinear system.

Such systems require iterative methods. We'll discuss Newton's method of solving the system and it's variations.

Newton's method for nonlinear fitting

Recall Newton's method for finding roots of equation $f(x) = 0$.

Let x_n be our current guess for the root $x^* = x_n + \Delta x$. Then Taylor expansion will be

$$0 = f(x^*) = f(x_n + \Delta x) = f(x_n) + \Delta x f'(x_n) + O((\Delta x)^2)$$

It follows that $f'(x_n)\Delta x \approx -f(x_n)$ which gives us update equation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Newton's method for nonlinear fitting

Recall Newton's method for finding roots of equation $f(x) = 0$.

Let x_n be our current guess for the root $x^* = x_n + \Delta x$. Then Taylor expansion will be

$$0 = f(x^*) = f(x_n + \Delta x) = f(x_n) + \Delta x f'(x_n) + O((\Delta x)^2)$$

It follows that $f'(x_n)\Delta x \approx -f(x_n)$ which gives us update equation

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

This argument generalizes directly to functions of several variables $F(x) = 0$

$$J_F(x_n)\Delta x_n = -F(x_n)$$

Newton's method for nonlinear fitting

In the case of nonlinear Least Squares we have $F(b) = J_r(b)^T r(b)$. To apply Newton's method we need to find the Jacobian of the function $F(b)$.

$$\begin{aligned}
\frac{\partial F_i}{\partial b_j} &= \frac{\partial}{\partial b_j}\left(J_r(b)^T r(b)\right)_i \\
&= \frac{\partial}{\partial b_j}\sum_{l=1}^{k}\frac{\partial r_l}{\partial b_i} r_l \\
&= \sum_{l=1}^{k}\frac{\partial r_l}{\partial b_i}\frac{\partial r_l}{\partial b_j} + \sum_{l=1}^{k}\frac{\partial^2 r_l}{\partial b_i \partial b_j} r_l
\end{aligned}$$

Newton's method for nonlinear fitting

Second derivatives are messy and painful to deal with. But they are multiplied by residuals which are small if function becomes a good fit. Therefore we can neglect second order term in the equality.

$$\frac{\partial F_i}{\partial b_j} \approx \sum_{l=1}^{k} \frac{\partial r_l}{\partial b_i} \frac{\partial r_l}{\partial b_j} = J_r(b)^T J_r(b)$$

Newton's method for nonlinear fitting

Second derivatives are messy and painful to deal with. But they are multiplied by residuals which are small if function becomes a good fit. Therefore we can neglect second order term in the equality.

$$\frac{\partial F_i}{\partial b_j} \approx \sum_{l=1}^{k} \frac{\partial r_l}{\partial b_i} \frac{\partial r_l}{\partial b_j} = J_r(b)^T J_r(b)$$

Putting all pieces together we obtain the update formula:

$$J_r(b_n)^T J_r(b_n) \Delta b_n = -J_r(b_n)^T r(b_n)$$

$$b_{n+1} = b_n + \Delta b_n$$

This is known as Gauss-Newton Algorithm for nonlinear Least Squares.

Every iteration requires solving a linear least squares problem
$J_r(b_n)\Delta b_n \approx -r(b_n)$

Computing the Jacobian

To use Gauss–Newton in practice, we need to be able to compute the Jacobian matrix $J_r(b_k)$ for any $b_k \in \mathbb{R}^n$

We can do this "by hand", *e.g.* in our transmitter/receiver problem we would have:

$$[J_r(b)]_{ij} = -\frac{\partial}{\partial b_j}\sqrt{(b_1 - x_1^i)^2 + (b_2 - x_2^i)^2}$$

Differentiating by hand is feasible in this case, but it can become impractical if $r(b)$ is more complicated

Or perhaps our mapping $b \to y$ is a "black box" — no closed form equations hence not possible to differentiate the residual!

Computing the Jacobian

So, what is the alternative to "differentiation by hand"?
Finite difference approximation: for $h \ll 1$ we have

$$[J_r(b_k)]_{ij} \approx \frac{r_i(b_k + e_j h) - r_i(b_k)}{h}$$

Avoids tedious, error prone differentiation of r by hand!

Also, can be used for differentiating "black box" mappings since we only need to be able to evaluate $r(b)$

Gauss-Newton with line search

Implementation of Gauss-Newton method often uses line search in the proposed direction of change:

$$b_{n+1} = b_n + \alpha_n \Delta b_n$$

Step length is chosen to satisfy Armijo condition

$$\phi(b_n + \alpha_n \Delta b_n) < \phi(b_n) + c\alpha_n r(b_n)^T J_r(b_n)^T \Delta b_n$$

for some constant $c \in (0, 1)$.

This provides "good enough" step in the descent direction. usual choice for α_n is the largest power on $1/2$ that satisfies the condition.

The Levenberg-Marquardt method

Levenberg-Marquardt methos is similar to Gauss-Newton method but line search is substituted with trust region strategy.

$$\min ||J_r(b_n)\Delta b_n + r(b_n)||^2 \text{ subject to } ||\Delta b_n|| \leq \text{ bound}$$

The Levenberg-Marquardt method

Levenberg-Marquardt methos is similar to Gauss-Newton method but line search is substituted with trust region strategy.

$$\min ||J_r(b_n)\Delta b_n + r(b_n)||^2 \text{ subject to } ||\Delta b_n|| \leq \text{ bound}$$

To solve this constrained optimization problem we need to optimize the following objective:

$$\min_{\Delta b_n} ||J_r(b_n)\Delta b_n + r(b_n)||^2 + \lambda_n ||\Delta b_n||^2$$

where λ_n is a Lagrange parameter for the constraint on n-th iteration.

The Levenberg-Marquardt method

The update step is computed as a solution to a Linear Least Squares problem

$$\min_{\Delta b_n} \left\| \begin{pmatrix} J_r(b_n) \\ \sqrt{\lambda_n} I \end{pmatrix} \Delta b_n - \begin{pmatrix} -r(b_n) \\ 0 \end{pmatrix} \right\|^2$$

Parameter λ_n influences both the direction and the length of the step. If λ_n is close to 0, we have Gauss-Newton method, for large λ_n we have a short step in the direction of steepest descent.

Common strategy for chosing λ_n is following:

1. The initial value is $\lambda_0 \approx ||J_r(b_0)^T J_r(b_0)||$
2. For subsequent steps improvement ratio ρ_n is defined as

$$\rho_n = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{\phi(b_n) - \phi(b_{n+1})}{\frac{1}{2}\Delta b_n^T (J_r(b_n)^T r(b_n) - \lambda_n \Delta b_n)}$$

The Levenberg-Marquardt method

Parameter λ_n influences both the direction and the length of the step. Common strategy for chosing λ_n is following:

1. The initial value is $\lambda_0 \approx ||J_r(b_0)^T J_r(b_0)||$
2. For subsequent steps improvement ratio ρ_n is defined as

$$\rho_n = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{\phi(b_n) - \phi(b_{n+1})}{\frac{1}{2}\Delta b_n^T (J_r(b_n)^T r(b_n) - \lambda_n \Delta b_n)}$$

- If $\rho_n > 0.75$ then $\lambda_{n+1} = \frac{\lambda_n}{3}$
- If $\rho_n < 0.25$ then $\lambda_{n+1} = 2\lambda_n$
- Otherwise $\lambda_{n+1} = \lambda_n$
- If $\rho_n > 0$ perform the update.

Both Gauss-Newton and Levenberg-Marquardt algorithms are often globally convergent with quadratic convergence if neglected second-order terms are small and linear otherwise.

The Levenberg-Marquardt method

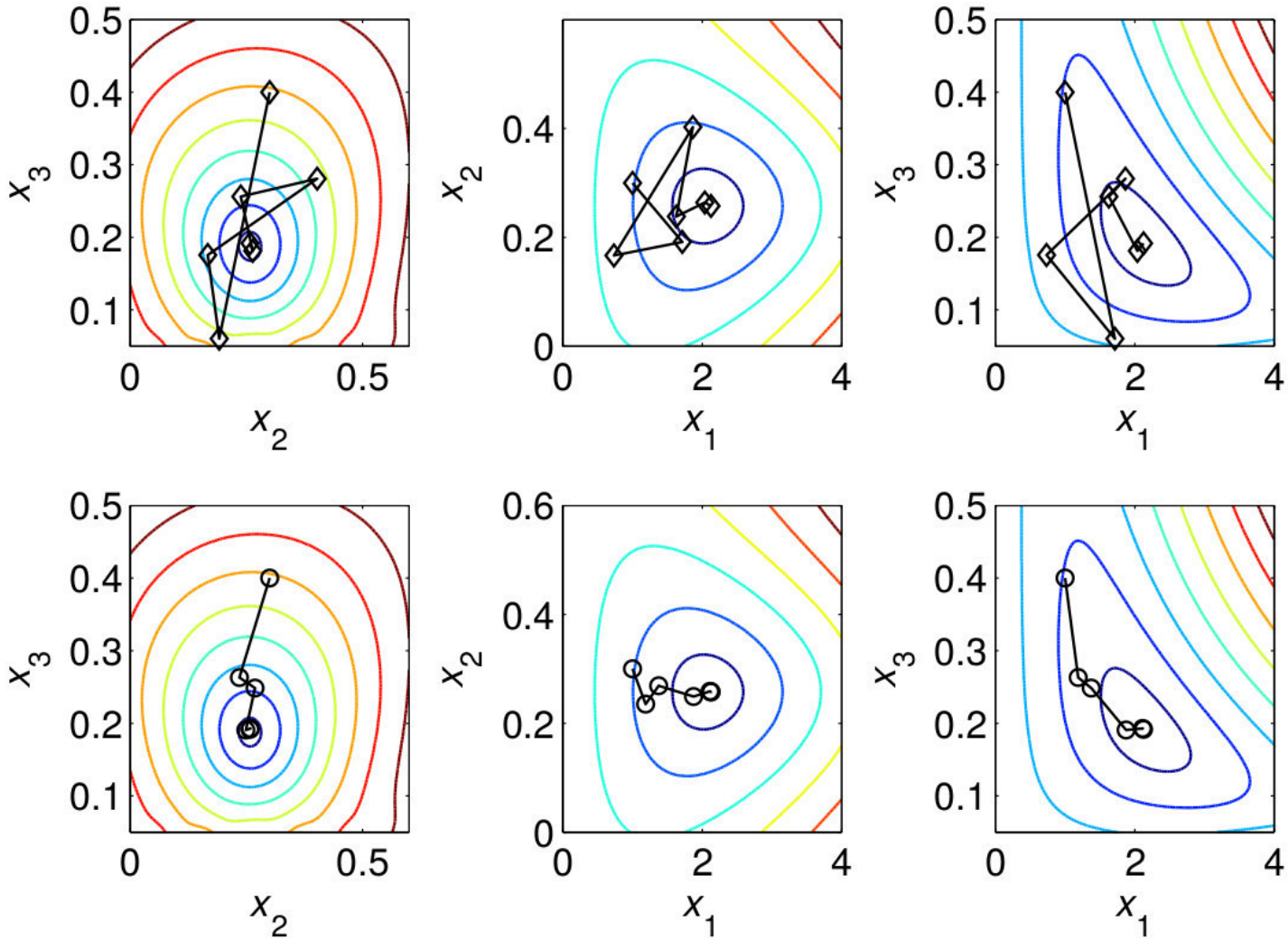
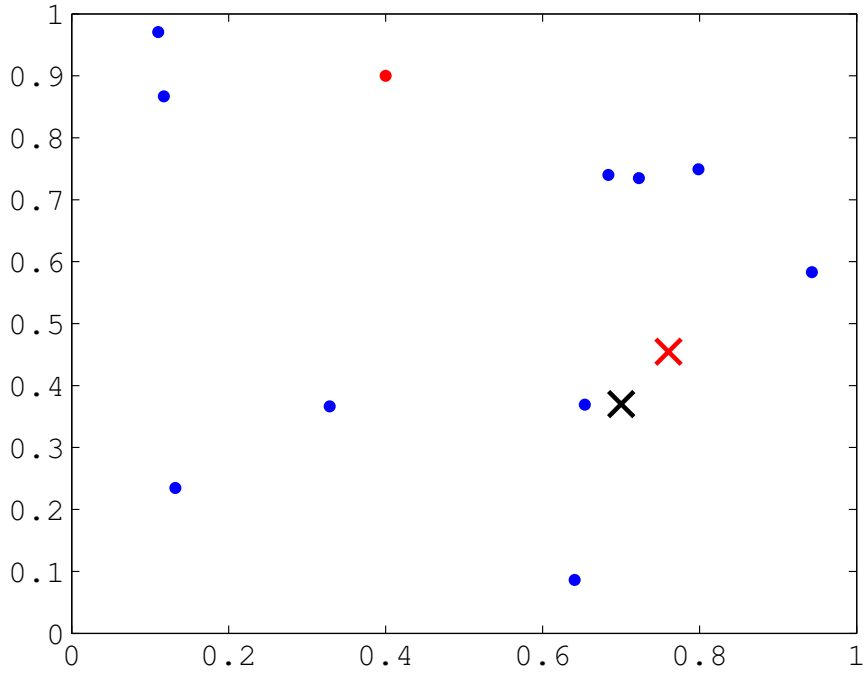


Figure: Gauss-Newton (top) and Levenberg-Marquardt (bottom) convergence

Nonlinear Least Squares: Example

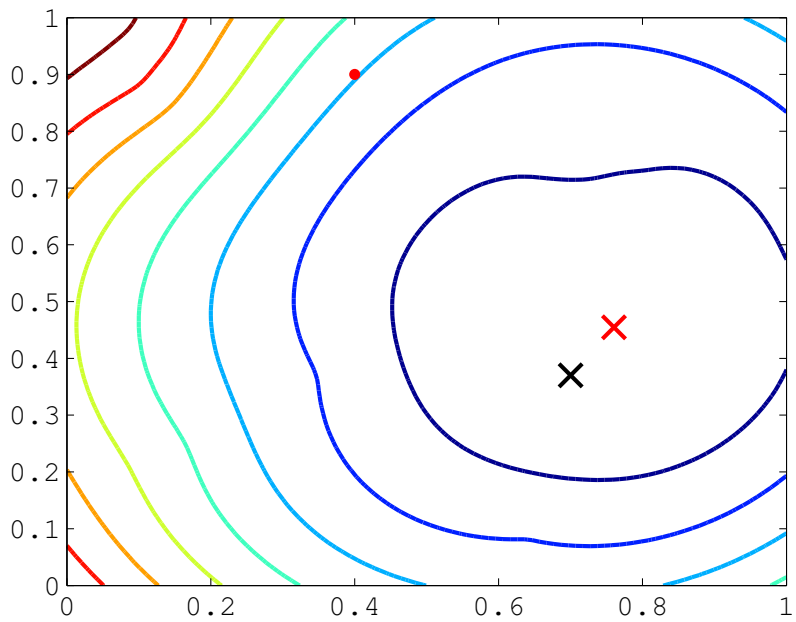
Python example: Using `lsqnonlin.py` we provide an initial guess (●), and converge to the solution (×)



Nonlinear Least Squares: Example

Levenberg–Marquardt minimizes $\phi(b)$, as we see from the contour plot of $\phi(b)$ below

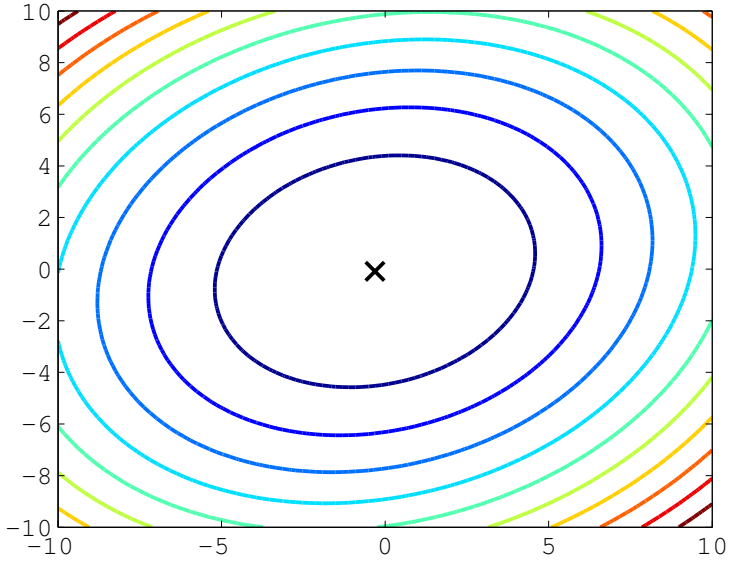
Recall $\times$ is the true transmitter location, $\times$ is our best-fit to the data; $\phi(\times) = 0.0248 < 0.0386 = \phi(\times)$.

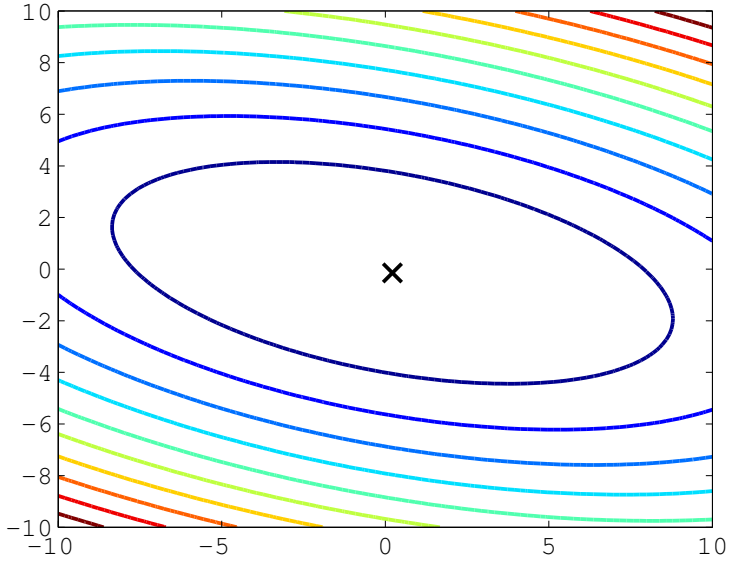


These contours are quite different from what we get in linear problems

Linear Least-Squares Contours

Two examples of linear least squares contours for $\phi(b) = \|y - Ab\|_2^2,\ b \in \mathbb{R}^2$





In linear least squares $\phi(b)$ is quadratic, hence contours are "hyperellipses"

Newton method for unconstrained minimization

$$\text{minimize} \quad f(x)$$

f convex, twice continously differentiable

Newton method

$$x^+ = x - t\nabla^2 f(x)^{-1}\nabla f(x)$$

- advantages: fast convergence, affine invariance
- disadvantages: requires second derivatives, solution of linear equation

can be too expensive for large scale applications

Variable metric methods

$$x^+ = x - tH^{-1}\nabla f(x)$$

$H \succ 0$ is approximation of the Hessian at x, chosen to:

- avoid calculation of second derivatives
- simplify computation of search direction

'Variable metric' interpretation (EE236B, lecture 10, page 11)

$$\Delta x = -H^{-1}\nabla f(x)$$

is steepest descent direction at x for quadratic norm

$$\|z\|_H = \left(z^T H z\right)^{1/2}$$

Quasi-Newton methods

given starting point $x^{(0)} \in \mathrm{dom}\, f$, $H_0 \succ 0$

1. compute quasi-Newton direction $\Delta x = -H_{k-1}^{-1} \nabla f(x^{(k-1)})$
2. determine step size t (*e.g.*, by backtracking line search)
3. compute $x^{(k)} = x^{(k-1)} + t\Delta x$
4. compute H_k

- different methods use different rules for updating H in step 4
- can also propagate H_k^{-1} to simplify calculation of Δx

Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

BFGS update

$$H_k = H_{k-1} + \frac{yy^T}{y^T s} - \frac{H_{k-1} s s^T H_{k-1}}{s^T H_{k-1} s}$$

where

$$s = x^{(k)} - x^{(k-1)}, \qquad y = \nabla f(x^{(k)}) - \nabla f(x^{(k-1)})$$

Inverse update

$$H_k^{-1} = \left(I - \frac{sy^T}{y^T s}\right) H_{k-1}^{-1} \left(I - \frac{ys^T}{y^T s}\right) + \frac{ss^T}{y^T s}$$

- note that $y^T s > 0$ for strictly convex f; see page 1-9
- cost of update or inverse update is $O(n^2)$ operations

Positive definiteness

if $y^T s > 0$, BFGS update preserves positive definitess of H_k

Proof: from inverse update formula,

$$v^T H_k^{-1} v = \left(v - \frac{s^T v}{s^T y} y\right)^T H_{k-1}^{-1} \left(v - \frac{s^T v}{s^T y} y\right) + \frac{(s^T v)^2}{y^T s}$$

- if $H_{k-1} \succ 0$, both terms are nonnegative for all v
- second term is zero only if $s^T v = 0$; then first term is zero only if $v = 0$

this ensures that $\Delta x = -H_k^{-1} \nabla f(x^k)$ is a descent direction

Secant condition

the BFGS update satisfies the *secant condition* $H_k s = y$, *i.e.*,

$$H_k(x^{(k)} - x^{(k-1)}) = \nabla f(x^{(k)}) - \nabla f(x^{(k-1)})$$

Interpretation: define second-order approximation at $x^{(k)}$

$$f_{\mathrm{quad}}(z) = f(x^{(k)}) + \nabla f(x^{(k)})^T(z - x^{(k)}) + \frac{1}{2}(z - x^{(k)})^T H_k(z - x^{(k)})$$

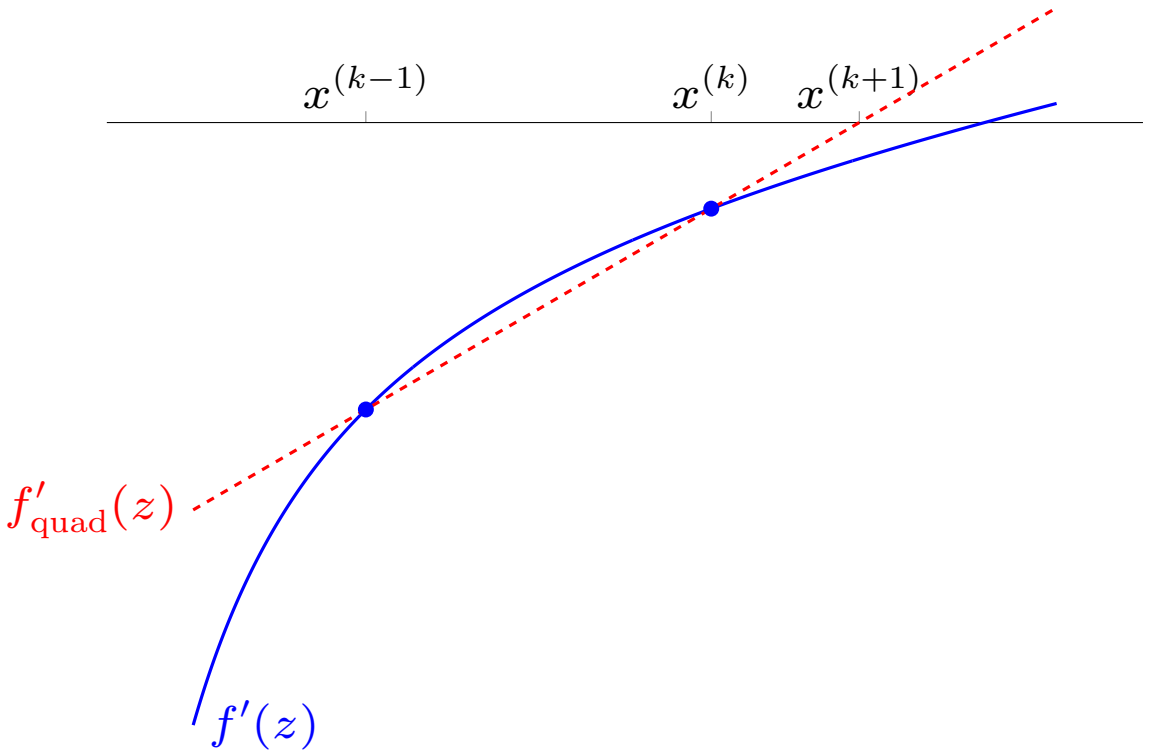
secant condition implies that gradient of f_{quad} agrees with f at $x^{(k-1)}$:

$$\begin{aligned} \nabla f_{\mathrm{quad}}(x^{(k-1)}) &= \nabla f(x^{(k)}) + H_k(x^{(k-1)} - x^{(k)}) \\ &= \nabla f(x^{(k-1)}) \end{aligned}$$

Secant method

for $f : \mathbf{R} \to \mathbf{R}$, BFGS with unit step size gives the secant method

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{H_k}, \qquad H_k = \frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$



Convergence

Global result

if f is strongly convex, BFGS with backtracking line search (EE236B, lecture 10-6) converges from any $x^{(0)}$, $H_0 \succ 0$

Local convergence

if f is strongly convex and $\nabla^2 f(x)$ is Lipschitz continuous, local convergence is *superlinear*: for sufficiently large k,

$$\|x^{(k+1)} - x^\star\|_2 \leq c_k \|x^{(k)} - x^\star\|_2 \to 0$$

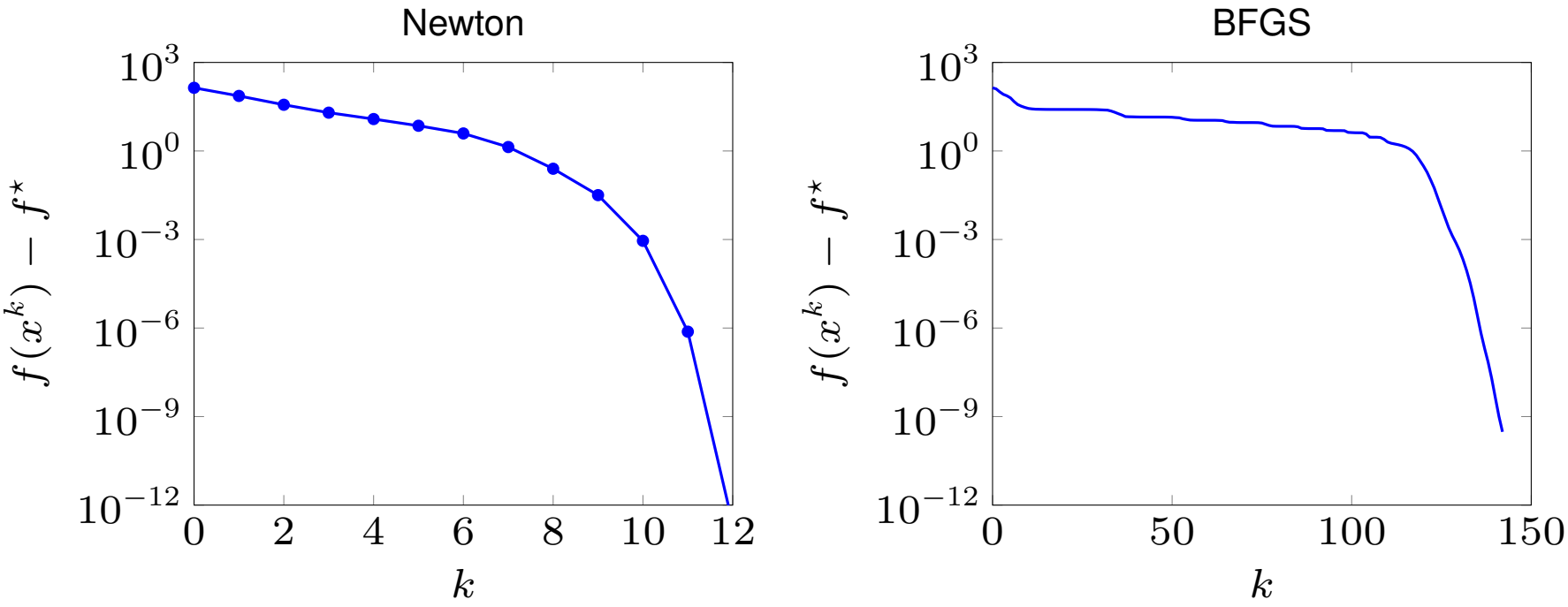
where $c_k \to 0$

(*cf.*, quadratic local convergence of Newton method)

Example

$$\text{minimize} \quad c^T x - \sum_{i=1}^{m} \log(b_i - a_i^T x)$$

$n = 100, m = 500$



- cost per Newton iteration: $O(n^3)$ plus computing $\nabla^2 f(x)$
- cost per BFGS iteration: $O(n^2)$

Limited memory quasi-Newton methods

main disadvantage of quasi-Newton method is need to store H_k or H_k^{-1}

Limited-memory BFGS (L-BFGS): do not store H_k^{-1} explicitly

- instead we store the m (*e.g.*, $m = 30$) most recent values of

$$s_j = x^{(j)} - x^{(j-1)}, \qquad y_j = \nabla f(x^{(j)}) - \nabla f(x^{(j-1)})$$

- we evaluate $\Delta x = H_k^{-1} \nabla f(x^{(k)})$ recursively, using

$$H_j^{-1} = \left(I - \frac{s_j y_j^T}{y_j^T s_j} \right) H_{j-1}^{-1} \left(I - \frac{y_j s_j^T}{y_j^T s_j} \right) + \frac{s_j s_j^T}{y_j^T s_j}$$

  for $j = k, k-1, \ldots, k-m+1$, assuming, for example, $H_{k-m}^{-1} = I$

- cost per iteration is $O(nm)$; storage is $O(nm)$