Optimization and algorithms

Module 2: unconstrained optimization

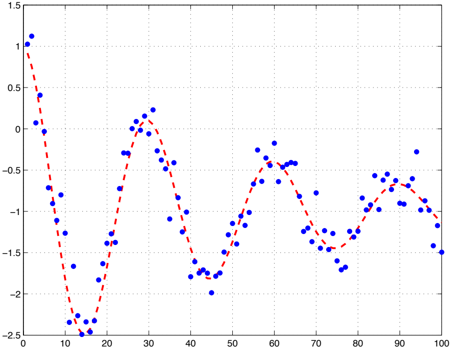
Team: João Xavier, João Sequeira, Hugo Pereira

Instituto Superior Técnico, Universidade de Lisboa

September 2024

Example: model fitting

- fit model $x(t) = a + be^{\sigma t}\cos(\omega t)$ to points $\{(t_p, x_p) \ : \ p = 1, \ldots, P\}$



- unconstrained optimization problem:

$$\underset{(a,b,\sigma,\omega)\in\mathbf{R}^4}{\text{minimize}} \quad \underbrace{\sum_{p=1}^{P} \left(a + be^{\sigma t_p}\cos(\omega t_p) - x_p\right)^2}_{f(a,b,\sigma,\omega)}$$

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad f(x)$$

- $x \in \mathbf{R}^n$ is the optimization variable
- $f : \mathbf{R}^n \to \mathbf{R}$ is the cost function
- $x^\star$ is a global minimum (also called a solution) if

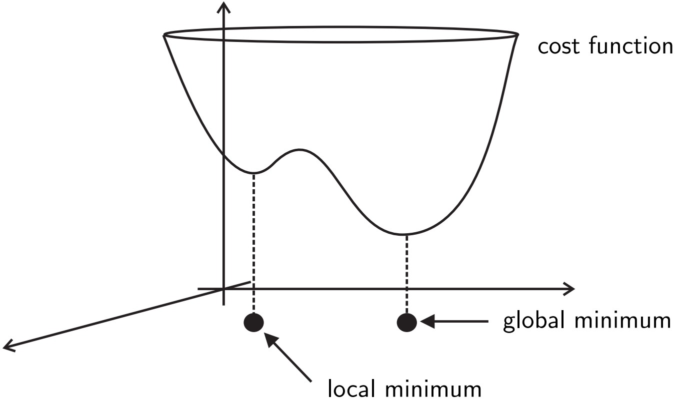
$$f(x^\star) \leq f(x), \quad \text{for all } x \in \mathbf{R}^n$$

- $x^\star$ is a local minimum if there exists $\epsilon > 0$ such that

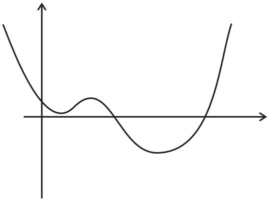
$$f(x^\star) \leq f(x), \quad \text{for all } x \in B(x^\star, \epsilon)$$

where

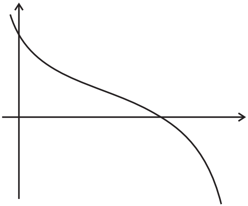
$$B(x^\star, \epsilon) = \{y : \|y - x\| < \epsilon\}$$


cost function
global minimum
local minimum

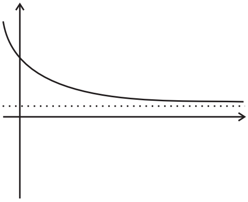
- the problem is said to be solvable if there exists a solution



solvable



not solvable

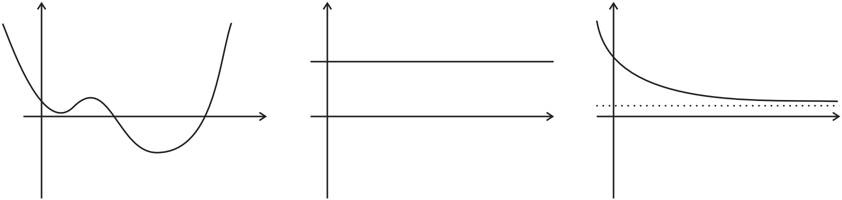


not solvable

When is a problem solvable?

- $f : \mathbf{R}^n \to \mathbf{R}$ is coercive if

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



coercive | not coercive | not coercive

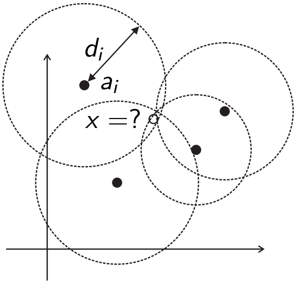
- **Theorem:** if f is coercive and continuous, then the problem

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

is solvable

Example: target localization

- x is unknown target position
- a_i is known position of ith sensor
- ith sensor measures $d_i = \|x - a_i\| + \text{noise}$



- where is the target?

- optimization problem

$$\underset{x \in \mathbf{R}^2}{\text{minimize}} \quad \underbrace{\sum_{i=1}^{m} \left(\|x - a_i\| - d_i\right)^2}_{f(x)}$$

is solvable because f is coercive and continuous

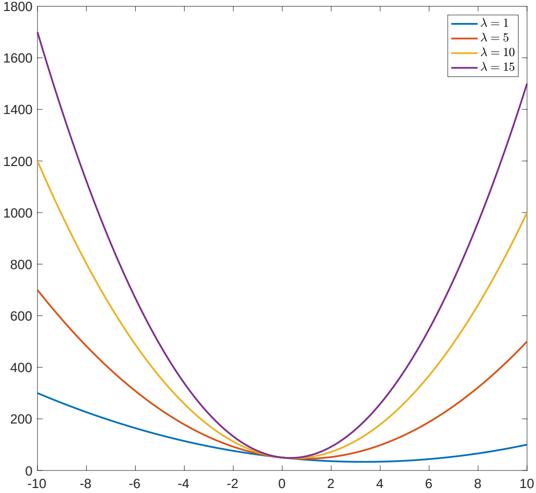
Example: the ℓ_2 "baby" denoising problem

- optimization problem

$$\underset{x \in \mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{2}(x-10)^2 + \lambda x^2}_{f(x)}$$

(where $\lambda > 0$) is solvable because f is coercive and continuous

The cost function f for several values of λ



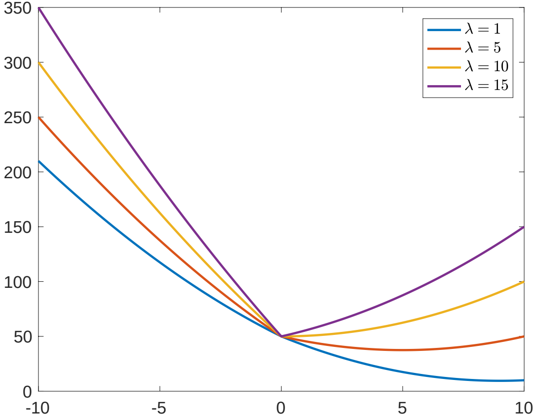
Example: the ℓ_1 "baby" denoising problem

- optimization problem

$$\underset{x \in \mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{2}(x-10)^2 + \lambda|x|}_{f(x)}$$

(where $\lambda > 0$) is solvable because f is coercive and continuous

The cost function f for several values of λ



How do we find solutions?

- **Theorem (first-order necessary condition):** if $x^\star$ is a local minimum and f is differentiable at $x^\star$, then

$$\nabla f(x^\star) = 0$$

  ($x^\star$ is a stationary point for f)

- finding the stationary points corresponds to solving the system

$$\begin{cases} \dfrac{\partial f}{\partial x_1}(x^\star) & = & 0 \\[2ex] \dfrac{\partial f}{\partial x_2}(x^\star) & = & 0 \\ & \vdots & \\ \dfrac{\partial f}{\partial x_n}(x^\star) & = & 0 \end{cases}$$

- in general, system is nonlinear and hard to solve
- theorem only gives us *candidates* to solutions

Example: solving the ℓ_2 "baby" denoising problem

- optimization problem is

$$\underset{x \in \mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{2}(x-10)^2 + \lambda x^2}_{f(x)}$$

(where $\lambda > 0$)

- we already know there is a solution, but where?

- because f is differentiable *everywhere*, a solution x must satisfy

$$\dot{f}(x) = 0 \quad \Leftrightarrow \quad x = 10/(1+2\lambda)$$

- conclusion: the solution *must* be $x = 10/(1+2\lambda)$

Example: solving the ℓ_1 "baby" denoising problem

- optimization problem is

$$\underset{x \in \mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{2}(x-10)^2 + \lambda|x|}_{f(x)}$$

(where $\lambda > 0$)

- we already know there is a solution, but where?

- because f is differentiable for $x > 0$, a solution $x > 0$ must satisfy

$$\begin{aligned} \dot{f}(x) = 0 \text{ and } x > 0 \quad &\Leftrightarrow \quad x = 10 - \lambda \text{ and } x > 0 \\ &\Leftrightarrow \quad x = 10 - \lambda \text{ and } \lambda < 10 \end{aligned}$$

- because f is differentiable for $x > 0$, a solution $x < 0$ must satisfy

$$\dot{f}(x) = 0 \text{ and } x < 0 \quad \Leftrightarrow \quad x = \lambda + 10 \text{ and } x < 0$$
$$\Leftrightarrow \quad x = \lambda + 10 \text{ and } \lambda < -10$$

(impossible, because $\lambda > 0$)

- conclusion:
  - for $0 < \lambda < 10$, the candidates for solutions are

    $$x = 10 - \lambda \quad \text{and} \quad x = 0.$$

    Which is a solution?
    Evaluating f on both, we find $f(10 - \lambda) < f(0)$; hence the solution is

    $$x = 10 - \lambda$$

  - for $\lambda > 10$, the only candidate for solutions is

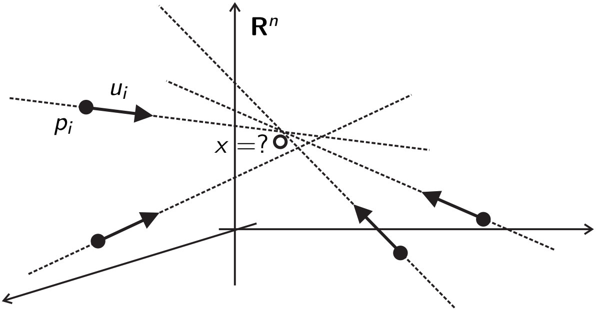
    $$x = 0;$$

    therefore, the solution *must* be

    $$x = 0$$

Example: locating an intruder

- m cameras point (with errors) to an intruder at unknown position $x \in \mathbf{R}^n$



- located at $p_i \in \mathbf{R}^n$, camera i points in the direction $u_i \in \mathbf{R}^n$ (with $\|u_i\| = 1$)

- what is the location x of the intruder?

- unconstrained optimization problem:

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{m} d(x, \mathcal{L}_i)^2$$

  where:

  - $\mathcal{L}_i = \{p_i + t u_i \;:\; t \in \mathbf{R}\}$ is the line spanned by camera i
  - $d(x, \mathcal{L}_i)$ is distance from x to line $\mathcal{L}_i$

- interpretation: we want the point x that deviates the least from all the camera lines

- note that

$$d(x, \mathcal{L}_i)^2 = (x - p_i)^T \Pi_i (x - p_i)$$

  where $\Pi_i := I - u_i u_i^T$

- equivalent optimization problem:

$$\underset{x}{\text{minimize}} \quad \underbrace{\frac{1}{2}x^T A x - b^T x}_{f(x)}$$

where $A = \sum_{i=1}^m \Pi_i$ and $b = \sum_{i=1}^m \Pi_i p_i$

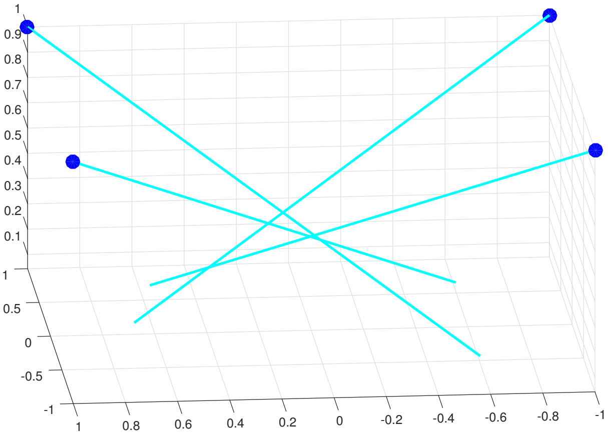
- finding the stationary points corresponds to solving a *linear* system

$$\nabla f(x) = 0 \quad \Leftrightarrow \quad Ax = b$$

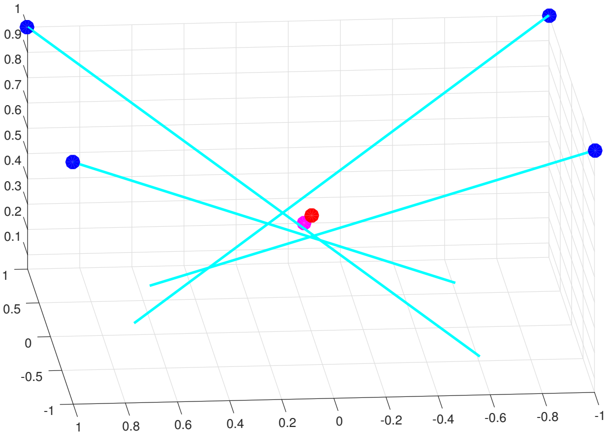
(in MATLAB, use `x = A\b`)

- important question: is a stationary point a global minimum? (to be answered...)

Example with $m = 4$ cameras:

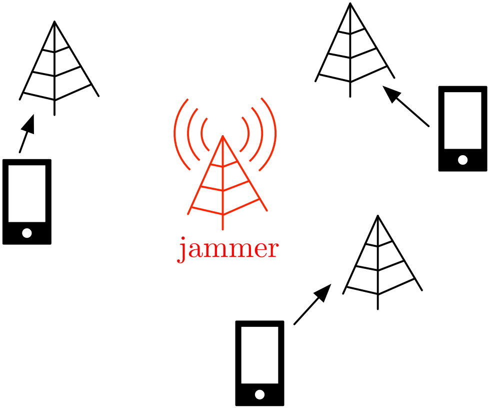


Example with $m = 4$ cameras:



True intruder position (red) and estimated position (magenta)

Example: silencing a jammer



- a powerful jammer disturbs $K + 1$ base stations

- base station $k = 0, 1, \ldots, K$, receives

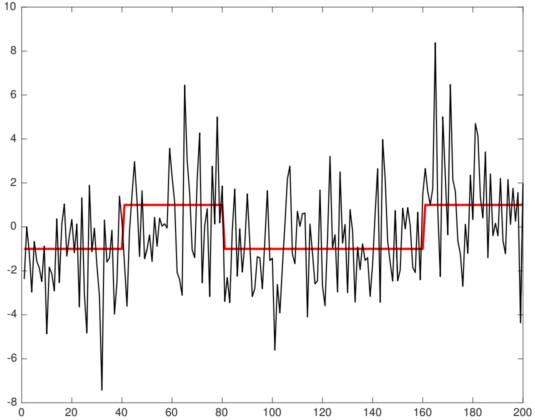
$$y_k = s_k + h_k w$$

where

  - $s_k \in \mathbf{R}$ is unknown signal from user i ($s_k^2 \simeq 1$)
  - $h_k \in \mathbf{R}$ is known channel from jammer to station k
  - $w \in \mathbf{R}$ is unknown signal from jammer ($w^2 \simeq P$ with P known)

- important: note that the jammer signal appears at all base stations

Example at base station 0 (segment with 200 signal samples):



s_0 (red) and $y_0 = s_0 + h_0 w$ (black)

- possible approach to mitigate interference at base station 0: combine linearly all measurements

$$\widehat{s}_0 \quad = \quad y_0 + c_1 y_1 + \cdots + c_K y_K$$

  where $c_1, \ldots, c_K$ are weights to chosen

- for a given choice of weights, we have

$$\widehat{s}_0 = s_0 + c_1 s_1 + \cdots + c_K s_K + (h_0 + c_1 h_1 + \cdots + c_K h_K) w$$

- we want $\widehat{s}_0 \simeq s_0$

- how do we choose the weights $c_1, \ldots, c_K$?

- since

$$\widehat{s}_0 = s_0 + c_1 s_1 + \cdots + c_K s_K + (h_0 + c_1 h_1 + \cdots + c_K h_K)w,$$

  we want to make

$$c_1 s_1 + \cdots + c_K s_K + (h_0 + c_1 h_1 + \cdots + c_K h_K)w \simeq 0$$

- but we don't know $s_1, \ldots, s_K, w$!

- possible approach: make

$$c_1^2 + \cdots + c_K^2 + (h_0 + c_1 h_1 + \cdots + c_K h_k)^2 P$$

  small

- leads to unconstrained optimization problem

$$\underset{c \in \mathbf{R}^K}{\text{minimize}} \quad \underbrace{\|c\|^2 + (h_0 + c^T h)^2 P}_{f(c)}$$

where

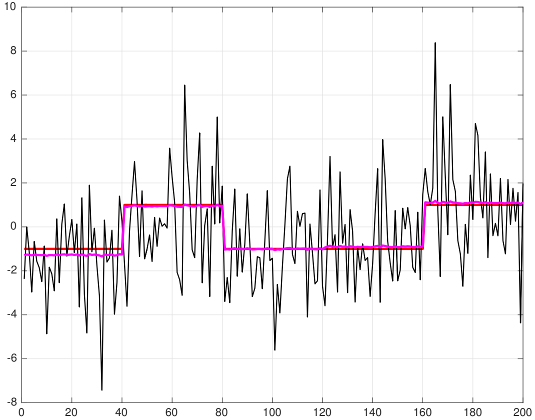
$$c = \begin{bmatrix} c_1 \\ \vdots \\ c_K \end{bmatrix} \quad \text{and} \quad h = \begin{bmatrix} h_1 \\ \vdots \\ h_K \end{bmatrix}$$

- finding the stationary points corresponds to solving a *linear* system

$$\nabla f(c) = 0 \quad \Leftrightarrow \quad (I + Phh^T)c = -h_0 Ph$$

- important question: is a stationary point a global minimum? (to be answered...)

Example at base station 0 (segment with 200 signal samples):



s_0 (red), $y_0 = s_0 + h_0 w$ (black), and $\widehat{s}_0 = y_0 + c_1 y_1 + \cdots + c_K y_K$ (magenta)

Numerical algorithms

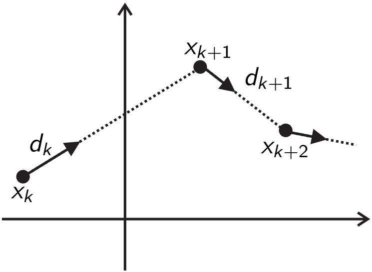
$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad f(x)$$

We will cover some standard algorithms for differentiable functions:

- gradient descent algorithm
- Newton algorithm
- Levenberg-Marquardt algorithm

Gradient descent algorithm

- gradient descent is an example of a *line search algorithm*



- in a line search algorithm, iterations evolve as

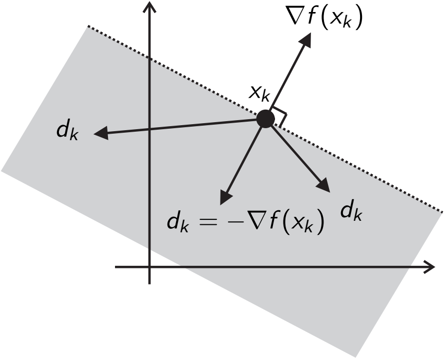
$$x_{k+1} = x_k + \alpha_k d_k, \qquad \text{for } k = 0, 1, 2, \dots$$

- $d_k \in \mathbf{R}^n$ is a descent direction for f at x_k:

$$\nabla f(x_k)^T d_k < 0$$

- $\alpha_k > 0$ is the step size

- infinite choices for descent directions d_k:



- **important fact**: if d_k is a descent direction for f at x_k, then there exists $\overline{\alpha} > 0$ such that

$$f(x_k + \alpha d_k) < f(x_k) \quad \text{for all } 0 < \alpha \leq \overline{\alpha}$$

Template for a line search algorithm

choose $x_0 \in \mathbf{R}^n$ and tolerance $\epsilon > 0$
set $k = 0$
loop
  compute $g_k = \nabla f(x_k)$
  check stopping criterion: if $\|g_k\| < \epsilon$ stop
  compute descent direction d_k
  compute step $\alpha_k > 0$
  update $x_{k+1} = x_k + \alpha_k d_k$
  $k \leftarrow k + 1$
end loop

- other stopping criteria can be used, e.g.,

$$\frac{|f(x_{k+1}) - f(x_k)|}{|f(x_k)| + 1} < \delta$$

- gradient descent algorithm uses $d_k = -\nabla f(x_k)$

- this d_k is a descent direction because

$$\nabla f(x_k)^T d_k = -\left\|\nabla f(x_k)\right\|^2 < 0$$

  (obviously, we assume x_k is not a stationary point; otherwise, we would be done!)

- the step size α_k is computed by the *backtracking* subroutine

Backtracking subroutine for computing the step size $\alpha_k > 0$

choose backtracking parameters $\widehat{\alpha} > 0$, $0 < \gamma < 0.5$, and $0 < \beta < 1$
set $\alpha_k := \widehat{\alpha}$
loop
  if $f\left(x_k + \alpha_k d_k\right) < f(x_k) + \gamma \nabla f(x_k)^T (\alpha_k d_k)$ stop
  $\alpha_k := \beta \alpha_k$
end loop

- Typical choices for the backtracking parameters:

$$\widehat{\alpha} = 1, \quad \gamma = 10^{-4}, \quad \beta = \frac{1}{2}$$

Gradient descent algorithm

choose $x_0 \in \mathbf{R}^n$ and tolerance $\epsilon > 0$
set $k = 0$
loop
  compute $g_k = \nabla f(x_k)$
  check stopping criterion: if $\|g_k\| < \epsilon$ stop
  set $d_k = -g_k$
  find $\alpha_k > 0$ with the backtracking subroutine
  update $x_{k+1} = x_k + \alpha_k d_k$
  $k \leftarrow k + 1$
end loop

Does the gradient descent algorithm converge?

- Let $(x_k)_{k\geq 0}$ be the sequence generated by the gradient descent algorithm

- **Theorem:** if f is a C^1 function and $x^\star$ is a limit point of $(x_k)_{k\geq 0}$, then
$$\nabla f\left(x^\star\right) = 0.$$

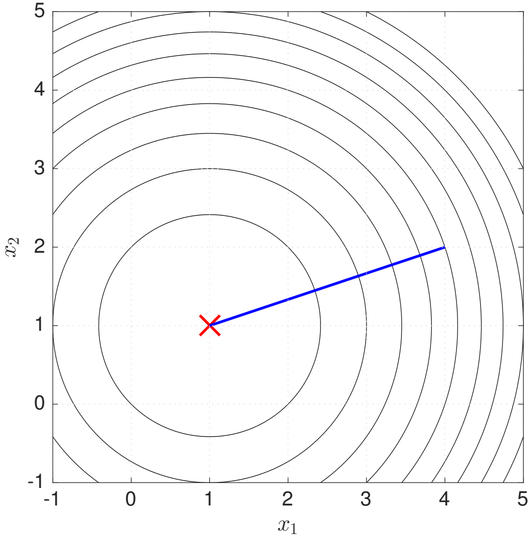
Example: toy quadratic in two dimensions

$$f : \mathbf{R}^2 \to \mathbf{R}, \quad f(x) = \frac{1}{2}(x-c)^T A(x-c)$$

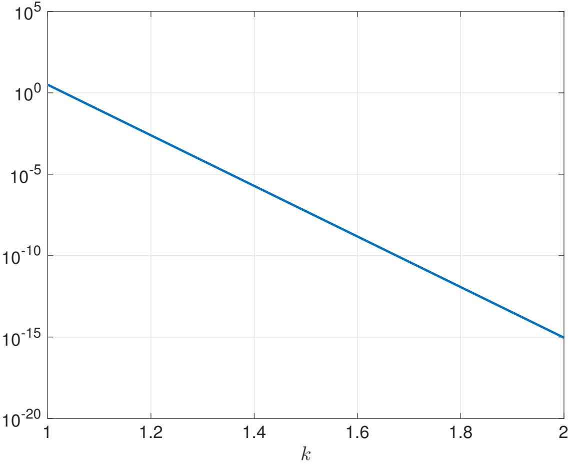
with

$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad A = Q \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} Q^T \qquad Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

- unique global minimum at $x^\star = c$
- note the condition number $\kappa\left(\nabla^2 f(x^\star)\right) = 1$
- parameters for gradient descent: $\epsilon = 10^{-6}$, $\widehat{\alpha} = 1$, $\gamma = 10^{-4}$, and $\beta = \frac{1}{2}$


5
4
3
2
1
0
-1
x_2
-1
0
1
2
3
4
5
x_1

$$\|\nabla f(x_k)\|$$



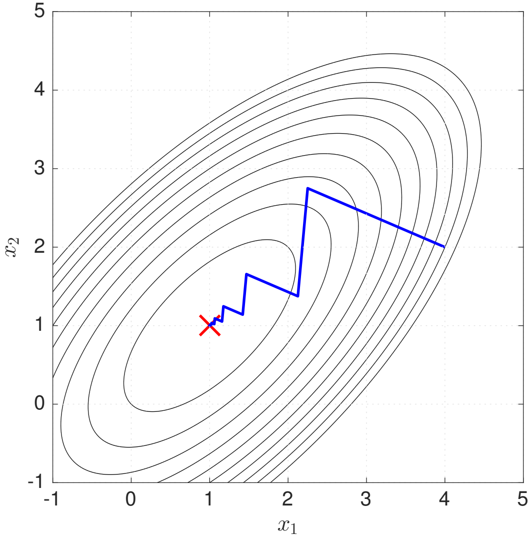
Example: toy quadratic in two dimensions

$$f \,:\, \mathbf{R}^2 \to \mathbf{R}, \quad f(x) = \frac{1}{2}(x-c)^T A(x-c)$$

with

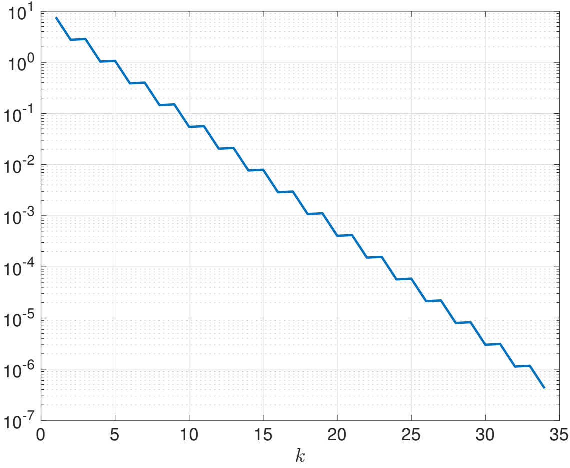
$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad A = Q \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} Q^T \qquad Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

- unique global minimum at $x^\star = c$
- note the condition number $\kappa\left(\nabla^2 f(x^\star)\right) = 5$
- parameters for gradient descent: $\epsilon = 10^{-6}$, $\widehat{\alpha} = 1$, $\gamma = 10^{-4}$, and $\beta = \frac{1}{2}$



5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
x_1
x_2

$\|\nabla f(x_k)\|$



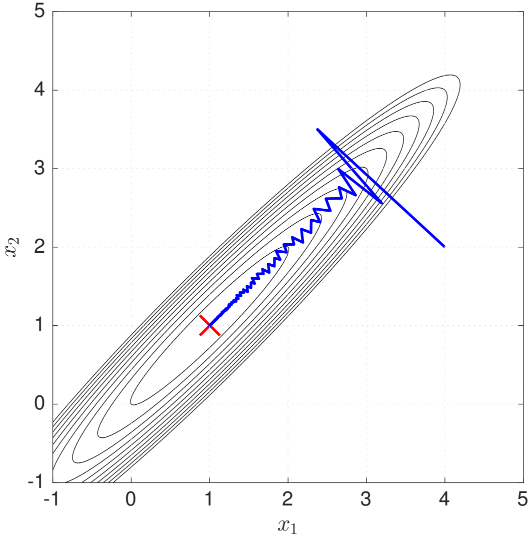
Example: toy quadratic in two dimensions

$$f : \mathbf{R}^2 \to \mathbf{R}, \quad f(x) = \frac{1}{2}(x-c)^T A (x-c)$$

with

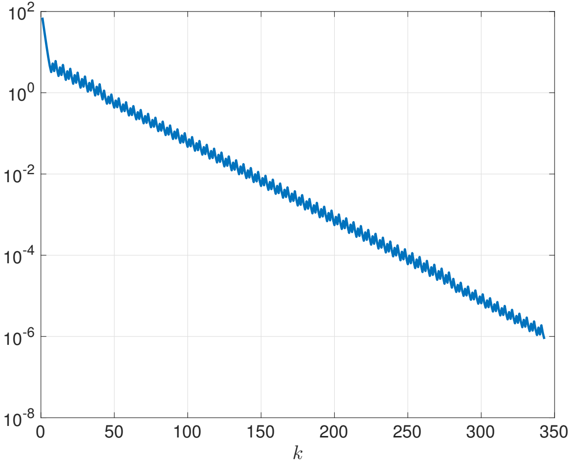
$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad A = Q \begin{bmatrix} 1 & 0 \\ 0 & 50 \end{bmatrix} Q^T \qquad Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

- unique global minimum at $x^\star = c$
- note the condition number $\kappa\left(\nabla^2 f(x^\star)\right) = 50$
- parameters for gradient descent: $\epsilon = 10^{-6}$, $\widehat{\alpha} = 1$, $\gamma = 10^{-4}$, and $\beta = \frac{1}{2}$



5
4
3
2
1
0
-1
x_2
-1
0
1
2
3
4
5
x_1

$\|\nabla f(x_k)\|$



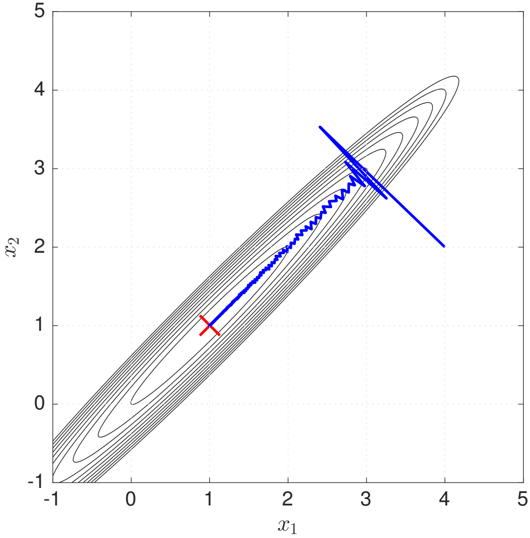
Example: toy quadratic in two dimensions

$$f\ :\ \mathbf{R}^2 \to \mathbf{R}, \quad f(x) = \frac{1}{2}(x-c)^T A(x-c)$$

with

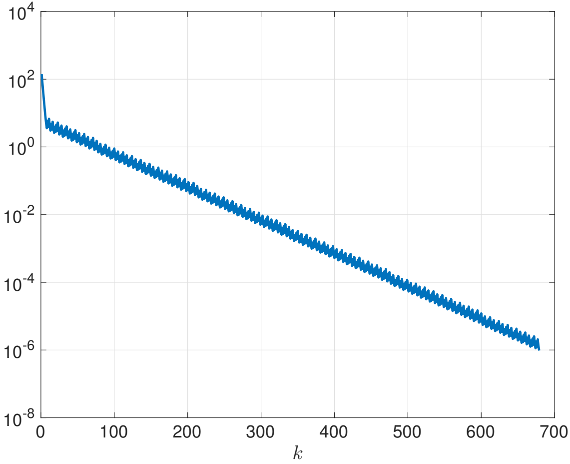
$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad A = Q \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix} Q^T \qquad Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

- unique global minimum at $x^\star = c$
- note the condition number $\kappa\left(\nabla^2 f(x^\star)\right) = 100$
- parameters for gradient descent: $\epsilon = 10^{-6}$, $\widehat{\alpha} = 1$, $\gamma = 10^{-4}$, and $\beta = \frac{1}{2}$



5
4
3
2
1
0
-1
-1
0
1
2
3
4
5
x_1
x_2

$$\|\nabla f(x_k)\|$$



Example: toy function in two dimensions

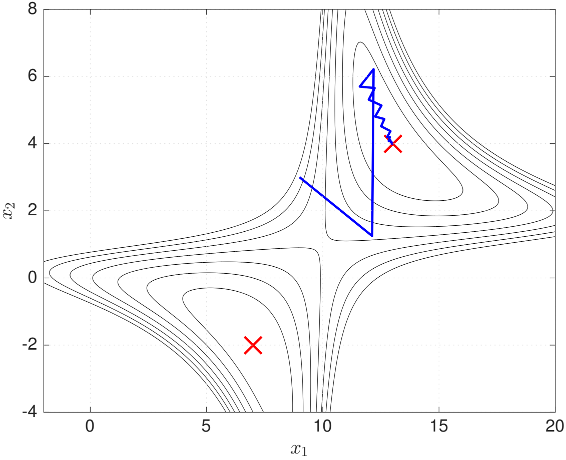
$$f : \mathbf{R}^2 \to \mathbf{R}, \quad f(x_1, x_2) = (11 - x_1 - x_2)^2 + (1 + x_1 + 10x_2 - x_1x_2)^2 - 40$$

- two global minima

  $$x^\star = \begin{bmatrix} 13 \\ 4 \end{bmatrix} \quad \text{and} \quad x^\star = \begin{bmatrix} 7 \\ -2 \end{bmatrix},$$

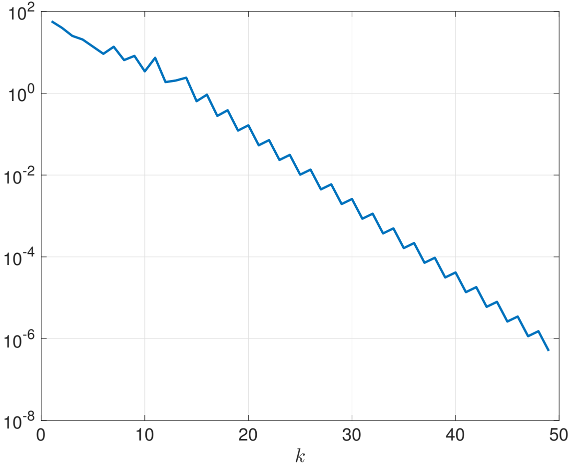
  both with $\kappa\left(\nabla^2 f(x^\star)\right) = 9$

- parameters for gradient descent: $\epsilon = 10^{-6}$, $\widehat{\alpha} = 1$, $\gamma = 10^{-4}$, and $\beta = \frac{1}{2}$



8
6
4
2
0
-2
-4
x_2
0
5
10
15
20
x_1

$\|\nabla f(x_k)\|$

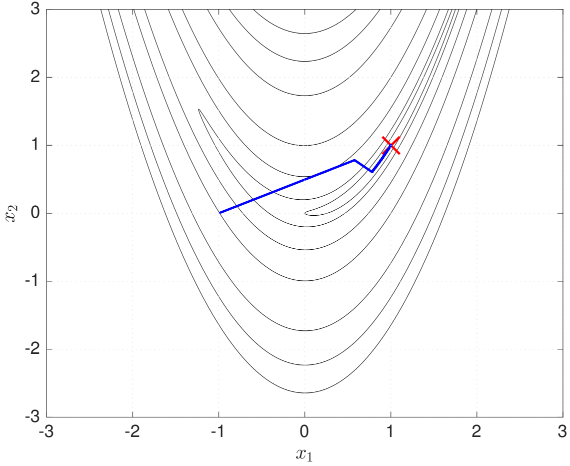


Example: toy function in two dimensions

$$f : \mathbf{R}^2 \to \mathbf{R}, \quad f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

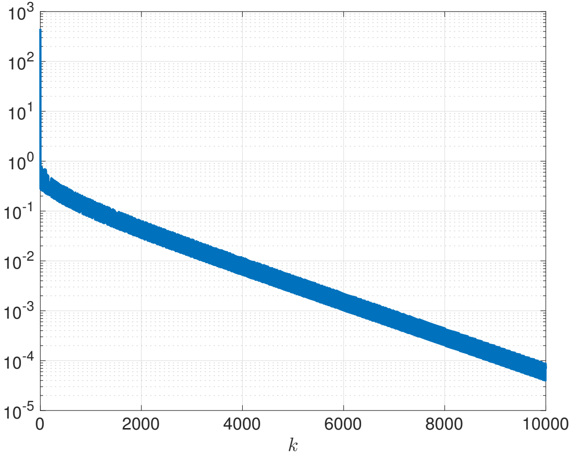
- unique global minimum
$$x^\star = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
with $\kappa\left(\nabla^2 f(x^\star)\right) = 2508$

- parameters for gradient descent: $\epsilon = 10^{-6}$, $\widehat{\alpha} = 1$, $\gamma = 10^{-4}$, and $\beta = \frac{1}{2}$



3
2
1
0
-1
-2
-3
x_2
-3
-2
-1
0
1
2
3
x_1

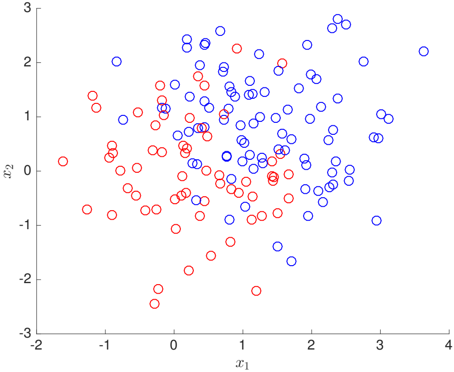
$\|\nabla f(x_k)\|$



Note: maximum number of iterations (10000) was attained before norm of gradient was less than $\epsilon = 10^{-6}$

Example: logistic regression

- dataset with M data points, labeled $m = 1, 2, \ldots, M$
- mth data point: $x_m \in \mathbf{R}^n$ (feature vector) and $y_m \in \{0, 1\}$ (class label)



- where do we draw the line separating the two classes?

- assume

$$\log \frac{\mathrm{Prob}(Y = 1 \,|\, X = x; s, r)}{\mathrm{Prob}(Y = 0 \,|\, X = x; s, r)} = s^T x - r$$

- $s \in \mathbf{R}^n$ and $r \in \mathbf{R}$ are (unknown) model parameters

- separating line is

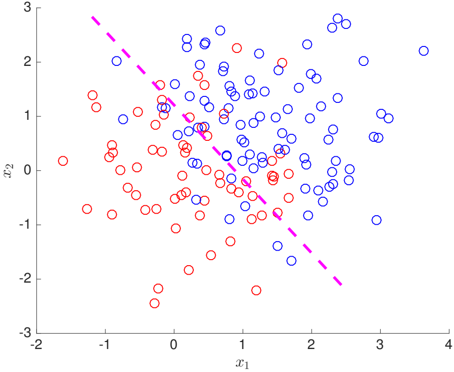
$$\left\{ x \in \mathbf{R}^n \;:\; s^T x = r \right\}$$

  (set of points for which the class labels are equally probable)

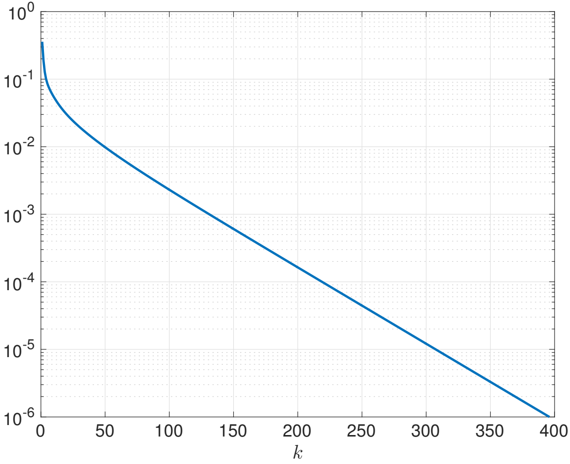
- discovering the separating line corresponds to finding the model parameters s and r

- finding the most likely model parameters leads to the optimization problem

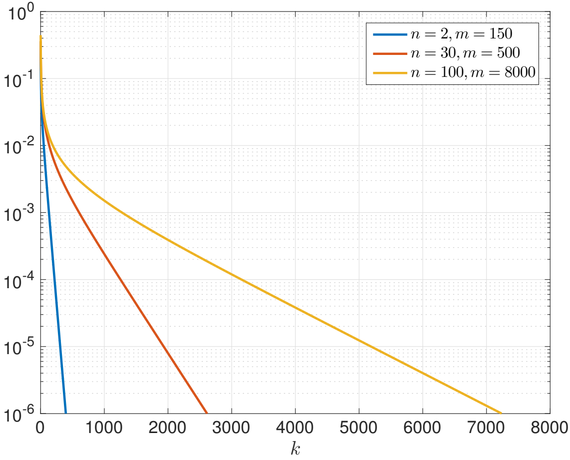
$$\underset{s\in\mathbf{R}^n, r\in\mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{K}\sum_{k=1}^{K} \log\left(1+\exp\left(s^T x_k - r\right)\right) - y_k\left(s^T x_k - r\right)}_{f(s,r)}$$



Norm of $\nabla f(x_k)$ along iterations



Norm of $\nabla f(x_k)$ along iterations



Classical Newton method

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad f(x)$$

- if $x^\star$ is a minimum then

$$\begin{cases} \dfrac{\partial f}{\partial x_1}(x^*) & = & 0 \\ \dfrac{\partial f}{\partial x_2}(x^*) & = & 0 \\ & \vdots & \\ \dfrac{\partial f}{\partial x_n}(x^*) & = & 0 \end{cases}$$

- how to solve the nonlinear system $\nabla f(x^\star) = 0$?

- consider system of nonlinear equations

$$\begin{cases} F_1(x_1, \dots, x_n) = 0 \\ F_2(x_1, \dots, x_n) = 0 \\ \qquad\vdots \\ F_n(x_1, \dots, x_n) = 0 \end{cases}$$

- more compactly: $F(x) = 0$ where $F : \mathbf{R}^n \to \mathbf{R}^n$, $F = (F_1, \dots, F_n)$

- classical Newton method from numerical analysis is

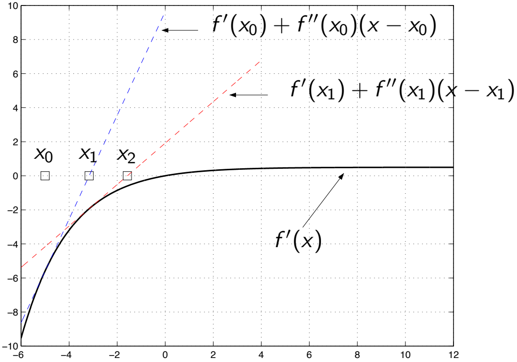
$$x_{k+1} = x_k - DF(x_k)^{-1} F(x_k)$$

- for $F(x) = \nabla f(x)$, classical Newton method is

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

- example: $f : \mathbf{R} \to \mathbf{R}$,

$$f(x) = e^{-\frac{1}{2}x} + \frac{1}{2}x$$



- classical Newton method

$$x_{k+1} = x_k - \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

does not converge in general (even for convex functions)

- it looks like a line search iteration

$$x_{k+1} = x_k + \alpha_k d_k$$

with $\alpha_k = 1$ and $d_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$

- key observation: if $\nabla^2 f(x_k) \succ 0$ (all eigenvalues are positive) then

$$\nabla f(x_k)^T d_k < 0,$$

that is, d_k is a descent direction

Newton algorithm (for convex functions)

1: choose $x_0 \in \mathbf{R}^n$ and tolerance $\epsilon > 0$
2: set $k = 0$
3: **loop**
4:     compute $g_k = \nabla f(x_k)$
5:     check stopping criterion: if $\|g_k\| < \epsilon$ stop
6:     set $d_k = -\left(\nabla^2 f(x_k)\right)^{-1} g_k$
7:     find $\alpha_k > 0$ with the backtracking subroutine
8:     update $x_{k+1} = x_k + \alpha_k d_k$
9:     $k \leftarrow k + 1$
10: **end loop**

- commonly, d_k is found in step 6 by solving the linear system

$$\nabla^2 f(x_k) d_k = -g_k$$

(for generic $n \times n$ Hessians, it takes $O(n^3)$ flops)

Does the Newton algorithm converge?

- Let $(x_k)_{k \geq 0}$ be the sequence generated by the Newton algorithm

- **Theorem:** if f is a C^3 function, $\nabla f(x^\star) = 0$, and $\nabla^2 f(x^\star)$ is positive definite, and x_0 is sufficiently close to $x^\star$, then

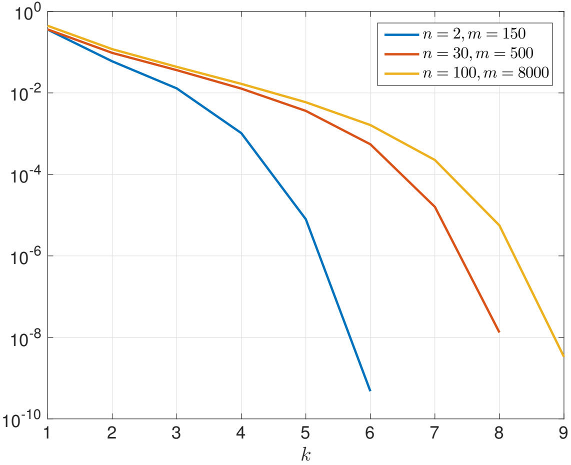
$$x_k \to x^\star \qquad \textit{at a quadratic rate}$$

- convergence at a quadratic rate means that there exists C and K such that

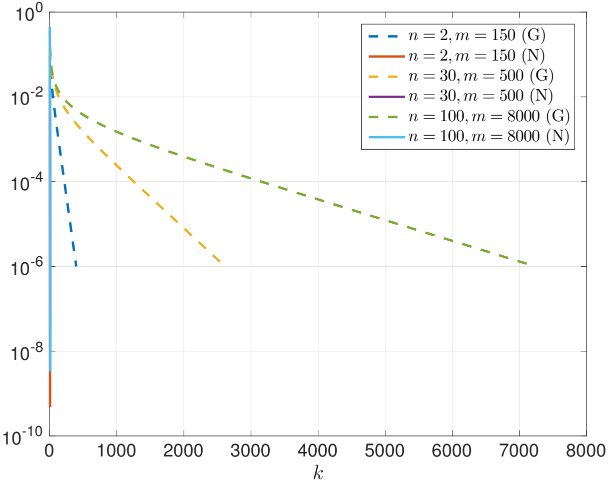
$$\|x_{k+1} - x^\star\| \leq C \|x_k - x^\star\|^2, \qquad \text{for all } k \geq K$$

(intuition: if $\|x_k - x^\star\| \simeq 10^{-d}$, then $\|x_{k+1} - x^\star\| \simeq 10^{-2d}$)

$\|\nabla f(x_k)\|$ along iterations



Gradient vs. Newton: $\|\nabla f(x_k)\|$ along iterations



The Levenberg-Marquardt (LM) algorithm

- LM algorithm addresses nonlinear least-squares problems:

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad \underbrace{f_1(x)^2 + f_2(x)^2 + \cdots + f_P(x)^2}_{f(x)},$$

  with differentiable functions $f_p : \mathbf{R}^n \to \mathbf{R}$ $(p = 1, 2, \ldots, P)$

- if all f_p's are affine functions, *i.e.*,

$$f_p(x) = a_p^T x - b_p,$$

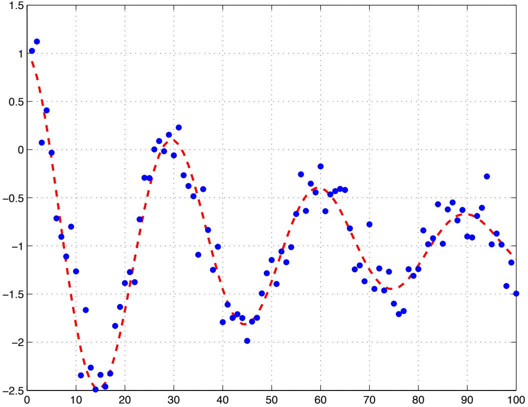
  we have a standard least-squares problem:

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad \textstyle\sum_{p=1}^{P} \left(a_p^T x - b_p\right)^2 = \left\| \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_P^T \end{bmatrix} x - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_P \end{bmatrix} \right\|^2$$

- example (from slide 2): fit signal model

$$x(t) = a + be^{\sigma t}\cos(\omega t)$$

to noisy measurements $\{(t_p, x_p) \,:\, p = 1, \ldots, P\}$



- optimization problem:

$$\underset{(a,b,\sigma,\omega)\in\mathbf{R}^4}{\text{minimize}} \quad \sum_{p=1}^{P} \left(a + be^{\sigma t_p}\cos(\omega t_p) - x_p\right)^2$$

- a nonlinear least-squares problem of the form

$$\underset{x\in\mathbf{R}^n}{\text{minimize}} \quad \sum_{p=1}^{P} f_p(x)^2,$$

with $x := (a, b, \sigma, \omega)$ and

$$f_p(a, b, \sigma, \omega) = a + be^{\sigma t_p}\cos(\omega t_p) - x_p$$

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad f_1(x)^2 + f_2(x)^2 + \cdots + f_P(x)^2$$

- LM algorithm generates a sequence $x_0, x_1, x_2, x_3, \ldots$ (x_0 is provided by the user)

- given x_k, LM starts by computing

$$\widehat{x}_{k+1} = \underset{x \in \mathbf{R}^n}{\operatorname{argmin}} \sum_{p=1}^{P} \left(f_p(x_k) + \nabla f_p(x_k)^T (x - x_k) \right)^2 + \lambda_k \left\| x - x_k \right\|^2$$

  for some $\lambda_k > 0$

- key points:
  - each function f_p is replaced by its linearization at x_k
  - λ_k penalizes deviations from x_k (rationale: linearization is good model only near x_k)

- the parameter λ_k is updated at each iteration

- if $f\left(\widehat{x}_{k+1}\right) < f(x_k)$, we made progress (linearization was a good model):
    - we accept the step: $x_{k+1} = \widehat{x}_{k+1}$
    - we decrease λ_k: $\lambda_{k+1} = 0.7\lambda_k$

- if $f\left(\widehat{x}_{k+1}\right) \geq f(x_k)$, we didn't made progress (linearization was not a good model):
    - we reject the step: $x_{k+1} = x_k$
    - we increase λ_k: $\lambda_{k+1} = 2\lambda_k$

LM algorithm (for nonlinear least-squares)

choose $x_0 \in \mathbf{R}^n$, $\lambda_0 > 0$, and tolerance $\epsilon > 0$

set $k = 0$

loop

  compute $g_k = \nabla f(x_k)$

  check stopping criterion: if $\|g_k\| < \epsilon$ stop

  solve

$$\widehat{x}_{k+1} = \underset{x \in \mathbf{R}^n}{\operatorname{argmin}} \sum_{p=1}^{P} \left(f_p(x_k) + \nabla f_p(x_k)^T (x - x_k)\right)^2 + \lambda_k \|x - x_k\|^2$$

  if $f\left(\widehat{x}_{k+1}\right) < f(x_k)$ [valid step]

$$x_{k+1} = \widehat{x}_{k+1}$$
$$\lambda_{k+1} = 0.7\lambda_k$$

  else [null step]

$$x_{k+1} = x_k$$
$$\lambda_{k+1} = 2\lambda_k$$

  $k \leftarrow k + 1$

end loop

- the optimization problem in step 6,

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \sum_{p=1}^{P} \left(f_p(x_k) + \nabla f_p(x_k)^T (x - x_k)\right)^2 + \lambda_k \left\|x - x_k\right\|^2,$$

is a standard least-squares problem

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \left\|Ax - b\right\|^2$$

where

$$A = \begin{bmatrix} \nabla f_1(x_k)^T \\ \nabla f_2(x_k)^T \\ \vdots \\ \nabla f_P(x_k)^T \\ \sqrt{\lambda_k} I \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} \nabla f_1(x_k)^T x_k - f_1(x_k) \\ \nabla f_2(x_k)^T x_k - f_2(x_k) \\ \vdots \\ \nabla f_P(x_k)^T x_k - f_P(x_k) \\ \sqrt{\lambda_k} x_k \end{bmatrix}$$

Example: toy function in two dimensions

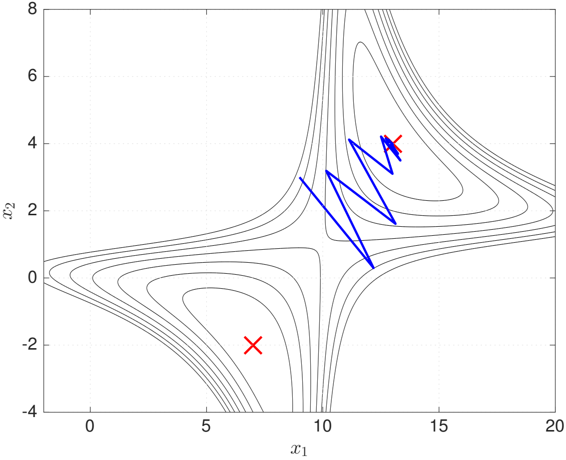
$$f : \mathbf{R}^2 \to \mathbf{R}, \quad f(x_1, x_2) = (11 - x_1 - x_2)^2 + (1 + x_1 + 10x_2 - x_1 x_2)^2 - 40$$

- two global minima

$$x^\star = \begin{bmatrix} 13 \\ 4 \end{bmatrix} \quad \text{and} \quad x^\star = \begin{bmatrix} 7 \\ -2 \end{bmatrix},$$

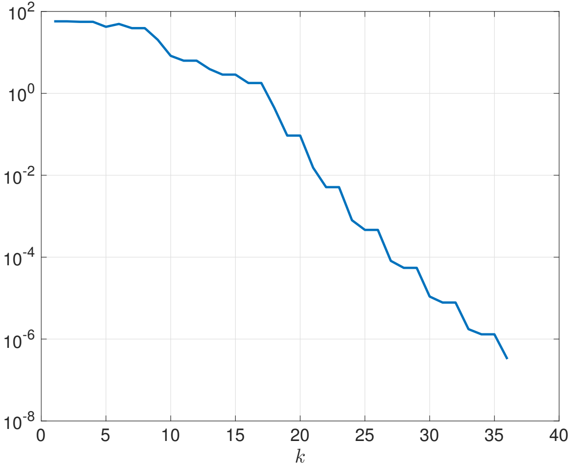
  both with $\kappa\left(\nabla^2 f(x^\star)\right) = 9$

- parameters for LM algorithm: $\epsilon = 10^{-6}$ and $\lambda_0 = 1$

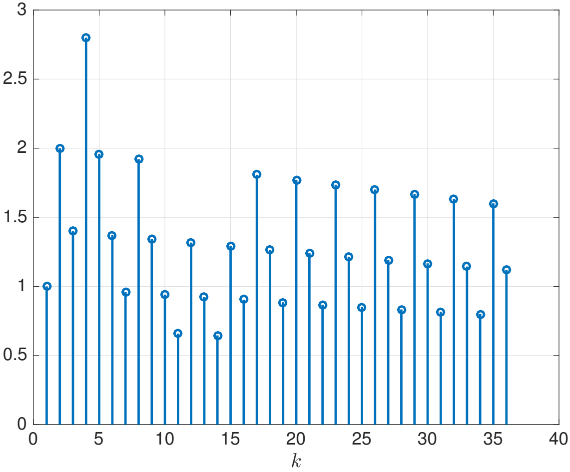


8
6
4
2
0
-2
-4
0
5
10
15
20
x_1
x_2

$$\|\nabla f(x_k)\|$$



λ_k



Example: toy function in two dimensions

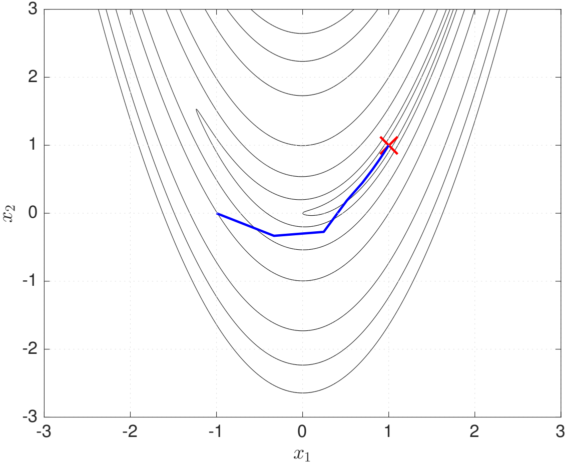
$$f\ :\ \mathbf{R}^2 \to \mathbf{R}, \quad f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- unique global minimum

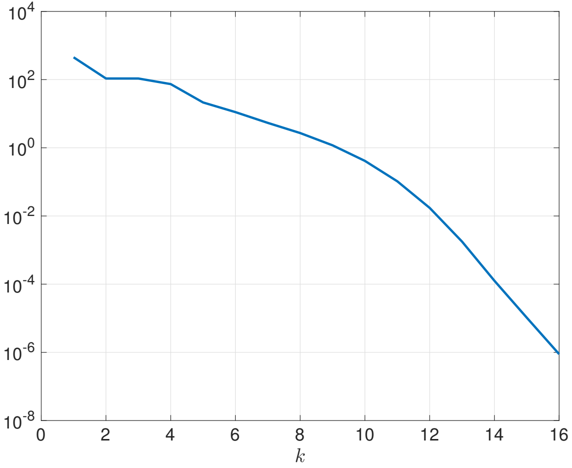
$$x^\star = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

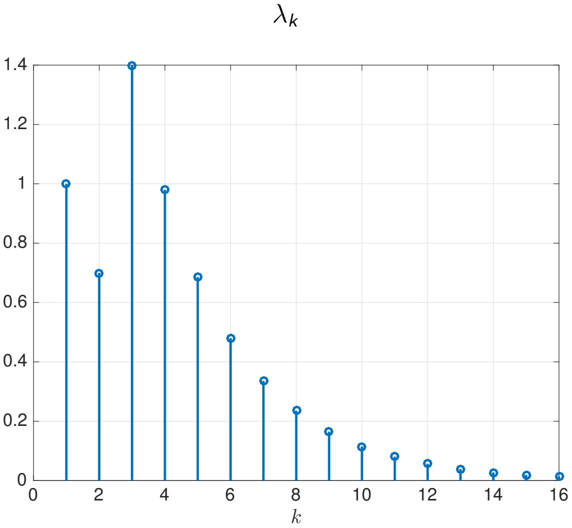
with $\kappa\left(\nabla^2 f(x^\star)\right) = 2508$

- parameters for LM algorithm: $\epsilon = 10^{-6}$ and $\lambda_0 = 1$


3
2
1
0
-1
-2
-3
x_2
-3
-2
-1
0
1
2
3
x_1

$$\|\nabla f(x_k)\|$$




λ_k
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
k