Lecture 3. Fitting data

Maksim Bolonkin

Moscow State University

What's the problem?

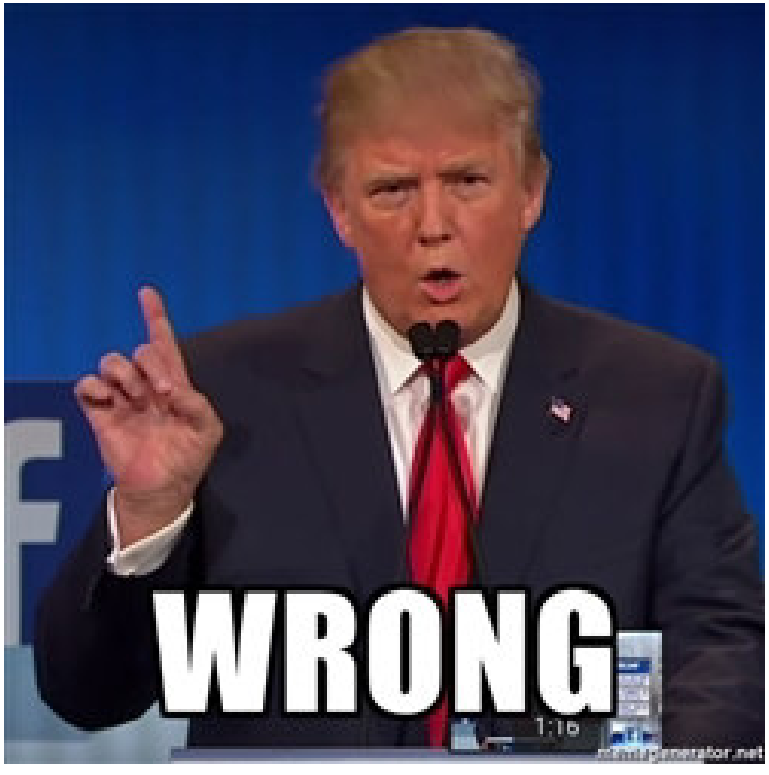
Given the dataset $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots (\boldsymbol{x_k}, y_k)\}$, $\boldsymbol{x_i} \in \mathbb{R}^m, y_i \in \mathbb{R}$ find a function $f : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(\boldsymbol{x_i}) = y_i, \quad i = 1, \ldots k$$

What's the problem?

Given the dataset $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots (\boldsymbol{x_k}, y_k)\}$, $\boldsymbol{x_i} \in \mathbb{R}^m, y_i \in \mathbb{R}$ find a function $f : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(\boldsymbol{x_i}) = y_i, \quad i = 1, \ldots k$$



What is the problem

Given the dataset $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots (\boldsymbol{x_k}, y_k)\}$, $\boldsymbol{x_i} \in \mathbb{R}^m, y_i \in \mathbb{R}$ find a function $f : \mathbb{R}^n \to \mathbb{R}$ such that

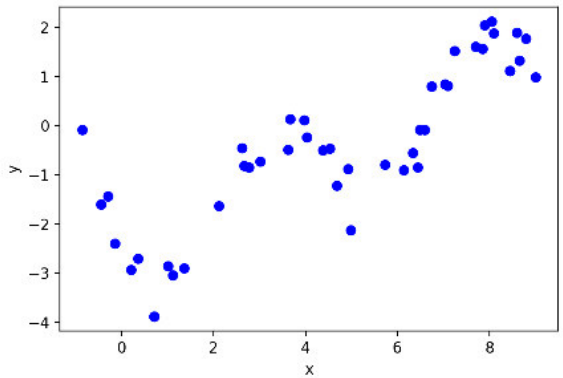
$$f(\boldsymbol{x_i}) \approx y_i, \quad i = 1, \ldots k$$

1. We want the function to be "good"
   - Continuous/differentiable function or mixture of such functions
2. We want the function to be a "good fit"
   - Depends on the measure of goodness: mean-squared error, log-likelihood, KL-divergence, etc.
   - **Data is always noisy**
3. We want the function to be generalizible
   - The dataset comes from some distribution
   - Function should be a "good fit" for out-of-sample data

Linear Least-Squares

We are considering a linear model

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \theta_i g_i(\boldsymbol{x})$$



Linear means linear in model parameters!

Functions
$g_i(\boldsymbol{x})$ reflect domain specific knowledge,
practical or computational considerations.

Examples: $x \in \mathbb{R}, y \in \mathbb{R}$

$$f(x, \boldsymbol{\theta}) = \theta_1 x + \theta_2$$

$$f(x, \boldsymbol{\theta}) = \theta_1 x^3 + \theta_2 x^2 + \theta_3 x + \theta_4$$

$$f(x, \boldsymbol{\theta}) = \theta_1 e^x + \theta_2 \sin x + \theta_3 x + \theta_4$$

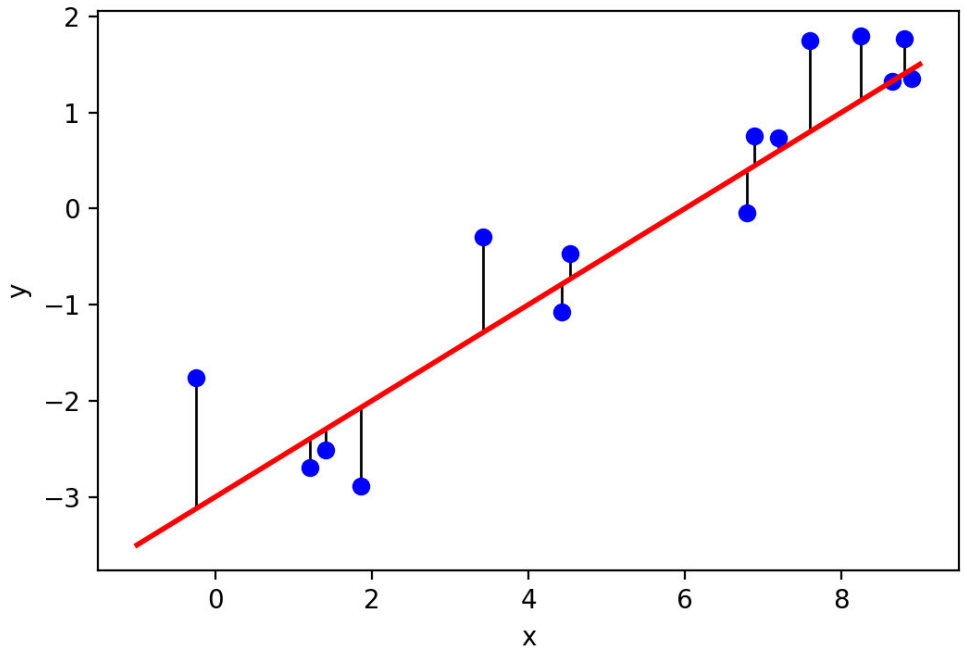
The Least Squares Fit

Let's consider **residuals** r_i associated with the data points:

$$r_i = y_i - f(\boldsymbol{x_i}, \theta)$$

We want to minimize the objective function:

$$\min_{\theta} \sum_{i=1}^{k} r_i(\boldsymbol{\theta})^2 = \min_{\theta} \sum_{i=1}^{k} (y_i - f(\boldsymbol{x_i}, \boldsymbol{\theta}))^2$$



Underlying ideas

We assume the following underlying data model:

$$y_i = \Gamma(\boldsymbol{x_i}) + \epsilon_i, \quad i = 1, 2, \ldots, k$$

Thus residuals can be expressed as following:

$$\begin{aligned} r_i &= y_i - f(\boldsymbol{x_i}, \boldsymbol{\theta}) \\ &= (y_i - \Gamma(\boldsymbol{x_i})) + (\Gamma(\boldsymbol{x_i}) - f(\boldsymbol{x_i}, \boldsymbol{\theta})) \\ &= \epsilon_i + (\Gamma(\boldsymbol{x_i}) - f(\boldsymbol{x_i}, \boldsymbol{\theta})) \end{aligned}$$

1. The *data error* ϵ_i comes from measurements
2. The *approximation error* $\Gamma(\boldsymbol{x_i}) - f(\boldsymbol{x_i}, \boldsymbol{\theta})$ is the difference between the pure-data function and the fitting model

Least Squares in Matrix Form

Let's define the matrix $A \in R^{k \times n}$

$$A = \begin{pmatrix} g_1(\boldsymbol{x}_1) & g_2(\boldsymbol{x}_1) & \dots & g_n(\boldsymbol{x}_1) \\ g_1(\boldsymbol{x}_2) & g_2(\boldsymbol{x}_2) & \dots & g_n(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\boldsymbol{x}_k) & g_2(\boldsymbol{x}_k) & \dots & g_n(\boldsymbol{x}_k) \end{pmatrix}$$

Let's also denote $y = (y_1 \ y_2 \ \dots \ y_k)^T$ a vector of observations. Then we have the following equations

$$A\theta \approx y, \quad r = y - A\theta$$

Least squares problem is

$$\min_{\theta} ||y - A\theta||^2$$

Closed-form solution

Function needs to be minimized $\longrightarrow$ derivative should be zero:

$$\frac{\partial}{\partial \theta}||y - A\theta||^2 = 0$$

Let's find the derivative:

$$\frac{\partial}{\partial \theta}||y - A\theta||^2 = -2A^T(y - A\theta) = 2A^TA\theta - 2A^Ty = 0$$

$$A^TA\theta = A^Ty$$

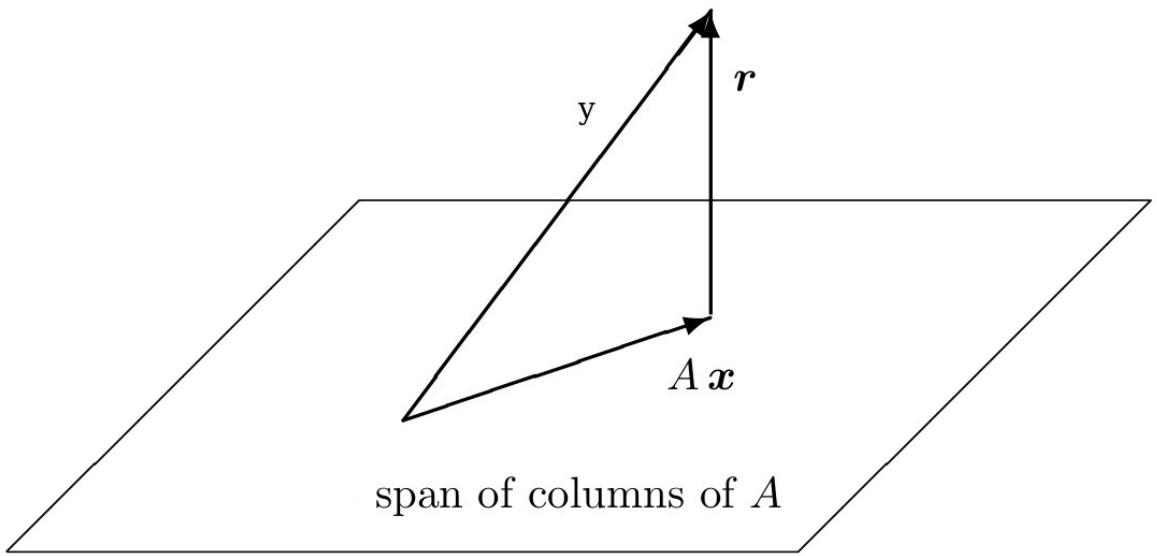
This is called the **normal equation**. Solution is:

$$\theta = (A^TA)^{-1}A^Ty$$

Normal equation

- Left-hand side and right-hand side of the insolvable equation $A\theta = y$ is multiplied by A^T
- If columns of A are independent then $A^T A$ is square, symmetric, and positive definite: matrix is invertable
- If columns of A are not independent then $A^T A$ is square, symmetric, and positive semi-definite: then need to use matrix factorization for inversion

Geometric interpretation:



Pseudoinverse matrix

Matrix $A^+ = (A^T A)^{-1} A^T$ is called *pseudoinverse* (Moore-Penrose pseudoinverse) and has the property $A^+ A = I$.

Pseudoinverse matrix is usually computed using one of the matrix decompositions.

Solving Least-Squares using LU decomposition

- Using LU decomposition for the matrix $A^T A$ and solving the normal equation using forward and backward eimination
- This approach is very unstable
- Condition number of $A^T A$ is the square of the condition number of A
- Condition number for positive definite matrix is the ratio of it's max and min eigenvalues

Solving Least-Squares using QR decomposition

- When stability is in doubt use orthogonalization $A = QR$
- In this case normal equation is getting simpler:

$$A^T A\theta = A^T y$$
$$(QR)^T QR\theta = (QR)^T y$$
$$R^T \underbrace{Q^T Q}_{I} R\theta = R^T Q^T y$$
$$R^T R\theta = R^T Q^T y$$
$$R\theta = Q^T y$$

- Multiplication $Q^T y$ is stable, back-substitution with upper-triangular R is simple
- Twice as long as for calculating $A^T A$ but much more reliable

Solving Least-Squares using SVD decomposition

- Most stable solution is using SVD decomposition $A = UDV^T$
- $A^T A = VD^T U^T UDV^T = VD^T DV^T = VD^2 V^T$
- In this case normal equation is getting simpler:

$$VD^2 V^T \theta = VDU^T y$$
$$D^2 V^T \theta = DU^T y$$
$$V^T \theta = \underbrace{(D^2)^{-1} D}_{D^+} U^T y$$
$$\theta = VD^+ U^T y$$

- If $rank(A) = n$ then

$$D^+ = D^{-1} = diag\left\{\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_n}\right\}$$

- Extremely small singular values can be removed.

Weighted Least Squares

Small but important extension of the least squares problem: not all the points are treated equally.

$$\min_{\theta} ||W(y - A\theta)||^2$$

Normal equation for θ is

$$(WA)^T(WA)\theta = (WA)^T y$$

No new math, just replace A with WA and y with Wy.

Total Least Squares

Another formulation of the least squares problem is following:

$$[\theta, \Delta y] = \arg\min_{\theta, \Delta y} ||\Delta y||, \text{ subject to } A\theta = y + \Delta y$$

In this form we are trying to modify y so that equation satisfies. Assumption is: observations are noisy, measurements are precise.

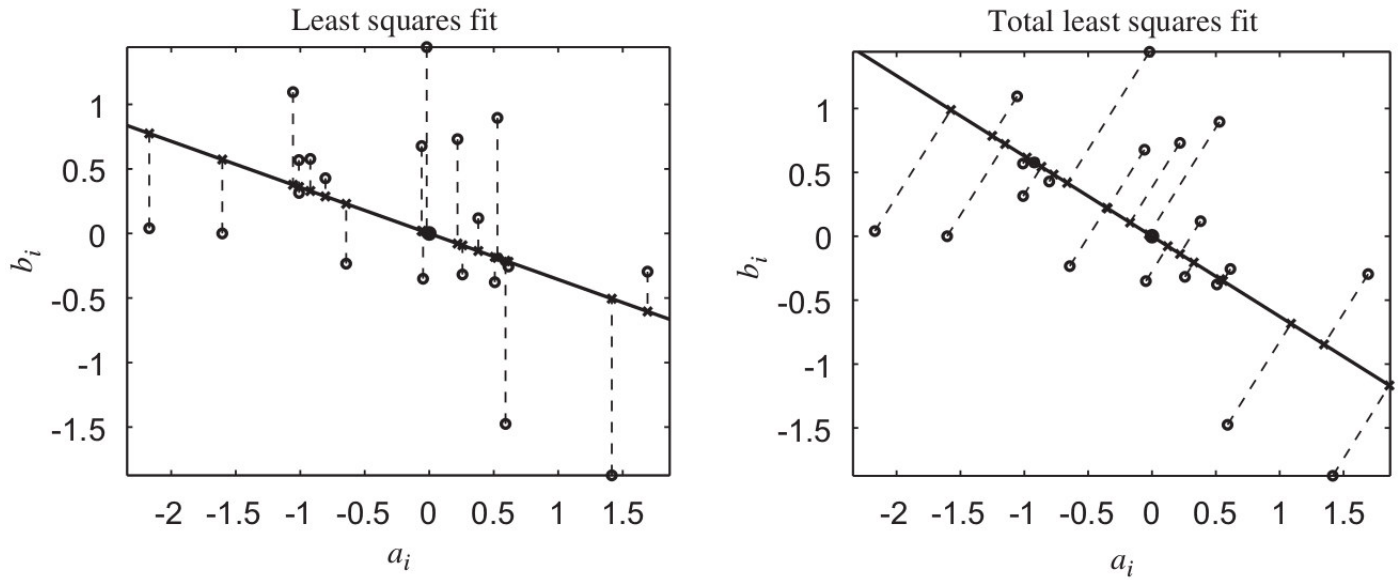
In case of noisy measurements we can consider the following problem:

$$[\theta, \Delta A, \Delta y] = \arg\min_{\theta, \Delta A, \Delta y} ||[\Delta A \;\; \Delta y]||, \text{ subject to } (A + \Delta A)\theta = y + \Delta y$$

This problem is called **total least squares**.

Total Least Squares

From geometric perspective total least squares minimizes sum of distances to the fitting line.



Total Least Squares

Let's consider 2-dimensional case. Assuming that all the points are centered around the origin (easy to obtain assumption), and line has a normal vector $(u_x, u_y)^T$.
In this case we have a following expression for distances:

$$Xu = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_k & y_k \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

We seek to minimize sum of squared distances:

$$\min_u ||Xu||^2, \text{ subject to } ||u|| = 1$$

Total Least Squares

$$\min_u ||Xu||^2, \text{ subject to } ||u|| = 1$$

This constrained optimization problem can be solved using Lagrange multiplier or as an unconstrained minimization of the Rayleigh quotient:

$$R(u) = \frac{||Xu||^2}{||u||^2} = \frac{u^T X^T X u}{u^T u}$$

As usually, function is being minimized by differentiating:

$$\frac{\partial R(u)}{\partial u} = \frac{(2X^T X u)(u^T u) - (u^T X^T X u)(2u)}{(u^T u)^2}$$

Equating enumerator to zero and getting:

$$(2X^T X u)(u^T u) = (u^T X^T X u)(2u)$$

$$X^T X u = \frac{u^T X^T X u}{u^T u} u$$

Total Least Squares

$$X^T X u = \frac{u^T X^T X u}{u^T u} u$$

Fraction on the right-hand side is exactly the Rayleigh quotient:

$$X^T X u = R(u) u$$

What does that mean?

1. Vector u is an eigenvector of a matrix $X^T X$ with eigenvalue $R(u)$
2. Since we want to minimize $R(u)$, the vector that minimize it is the eigenvector of $X^T X$ with minimal eigenvalue

Total Least Squares: general solution

Let's consider the matrix $C = [A \;\; y]$ and find SVD-decomposition of this matrix

$$C = [A \;\; y] = U\Sigma V^T, \text{ where } \Sigma = diag\{\sigma_1, \sigma_2, \ldots, \sigma_{n+1}\}$$

and define the partitioning

$$V = \begin{pmatrix} V_{11} & \boldsymbol{v}_{12} \\ \boldsymbol{v}_{21} & v_{22} \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \sigma_{22} \end{pmatrix}$$

Solution exists if $\sigma_{22} \neq 0$ and is given by

$$\theta = -\frac{\boldsymbol{v}_{12}}{v_{22}}$$

Underdetermined Least Squares

So far we focused only on overconstrained systems (more data points than parameters).
But Least Squares also applies to underconstrained systems $A\theta = y$ with $A \in \mathbb{R}^{k \times n}, k < n$

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} & \\ b & \\ & \end{bmatrix} = \begin{bmatrix} \\ y \\ \end{bmatrix}$$

We can apply the same math as before, but $A^T A$ will be singular, this infinitely many solutions.

Underdetermined Least Squares

One option is to pose the problem as constrained optimization

$$\min \theta^T \theta \text{ subject to } A\theta = y$$

Then use Lagrange multipliers method.

The idea is to restrict the solution to $(n-k)$-dimensional hyperplane of $\mathbb{R}^n$ on which $\theta^T\theta$ has a unique minimum.

Underdetermined Least Squares

One option is to pose the problem as constrained optimization

$$\min \theta^T \theta \text{ subject to } A\theta = y$$

Then use Lagrange multipliers method.

The idea is to restrict the solution to $(n-k)$-dimensional hyperplane of $\mathbb{R}^n$ on which $\theta^T\theta$ has a unique minimum.

Lagrange multiplier approach gives the solution

$$\theta = A^T(AA^T)^{-1}y$$

In this expression $A^T(AA^T)^{-1}$ is called right pseudo-inverse.

Underdetermined Least Squares

Another approach to the underconstrained problem is to impose regularization by modifying objective function:

$$\min ||y - A\theta||^2 + ||S\theta||^2$$

Here $S \in \mathbb{R}^{n \times n}$ is a scaling matrix.

Underdetermined Least Squares

Another approach to the underconstrained problem is to impose regularization by modifying objective function:

$$\min ||y - A\theta||^2 + ||S\theta||^2$$

Here $S \in \mathbb{R}^{n \times n}$ is a scaling matrix.

Calculating gradient of objective function gives the normal equation

$$(A^T A + S^T S)\theta = A^T y$$

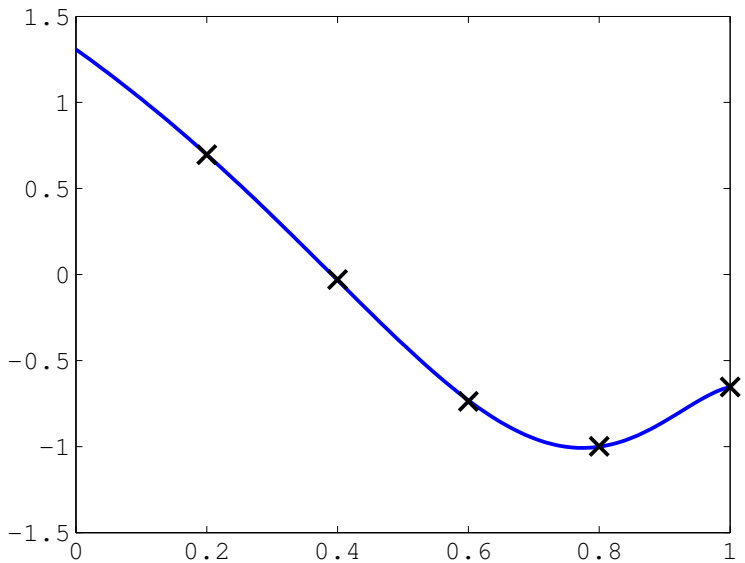
We chose S to ensure that lef-hand side matrix is invertible.

Simplest choice for the regularizer is $S = \lambda I \in \mathbb{R}^{n \times n}, \lambda > 0$

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = 0.001\mathrm{I}$ (*i.e.* $\mu = 0.001$)



$$\|r(b)\|_2 = 1.07 \times 10^{-4}$$

$$\|b\|_2 = 4.40$$

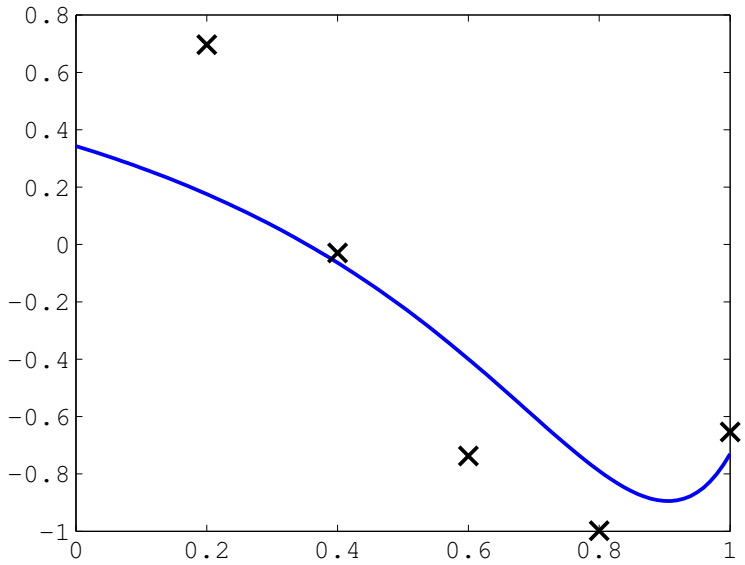
$$\mathrm{cond}(A^T A + S^T S) = 1.54 \times 10^7$$

Fit is good since regularization term is small but condition number is still large

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = 0.5\mathrm{I}$ (*i.e.* $\mu = 0.5$)



$$\|r(b)\|_2 = 6.60 \times 10^{-1}$$

$$\|b\|_2 = 1.15$$

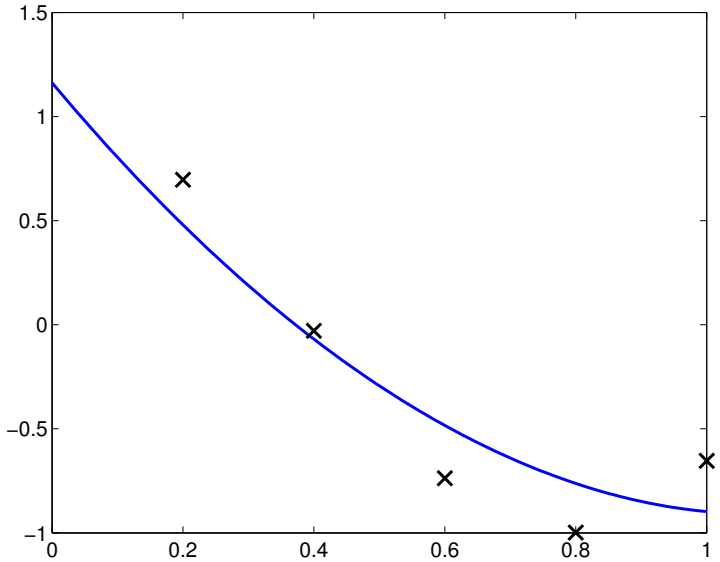
$$\mathrm{cond}(A^T A + S^T S) = 62.3$$

Regularization term now dominates: small condition number and small $\|b\|_2$, but poor fit to the data!

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = \texttt{diag}(0.1, 0.1, 0.1, 10, 10 \ldots, 10)$



$$\|r(b)\|_2 = 4.78 \times 10^{-1}$$

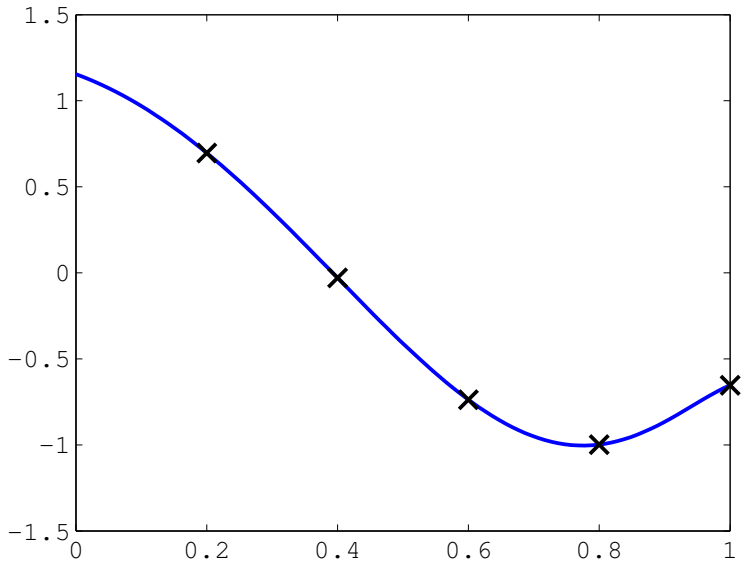
$$\|b\|_2 = 4.27$$

$$\text{cond}(A^T A + S^T S) = 5.90 \times 10^3$$

We strongly penalize $b_3, b_4, \ldots, b_{11}$, hence the fit is close to parabolic

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$



$$\|r(b)\|_2 = 1.03 \times 10^{-15}$$

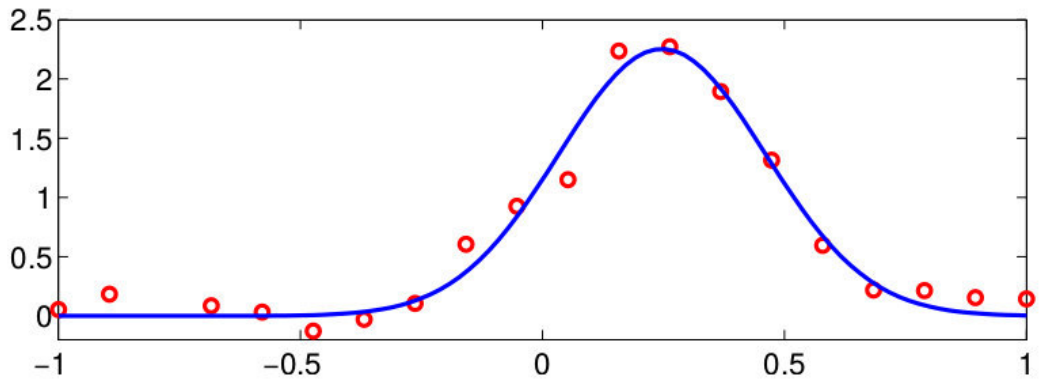
$$\|b\|_2 = 7.18$$

Python routine gives Lagrange multiplier based solution, hence satisfies the constraints to machine precision

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 $M(x, \boldsymbol{c}) = c_1 e^{-(x-c_2)^2/c_3^2}$

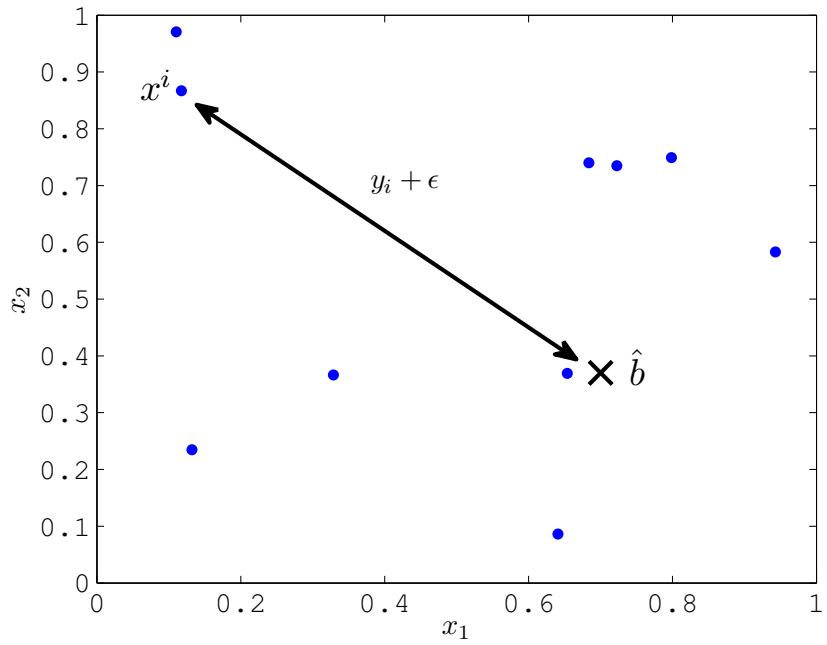


Nonlinear Least Squares: Example

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

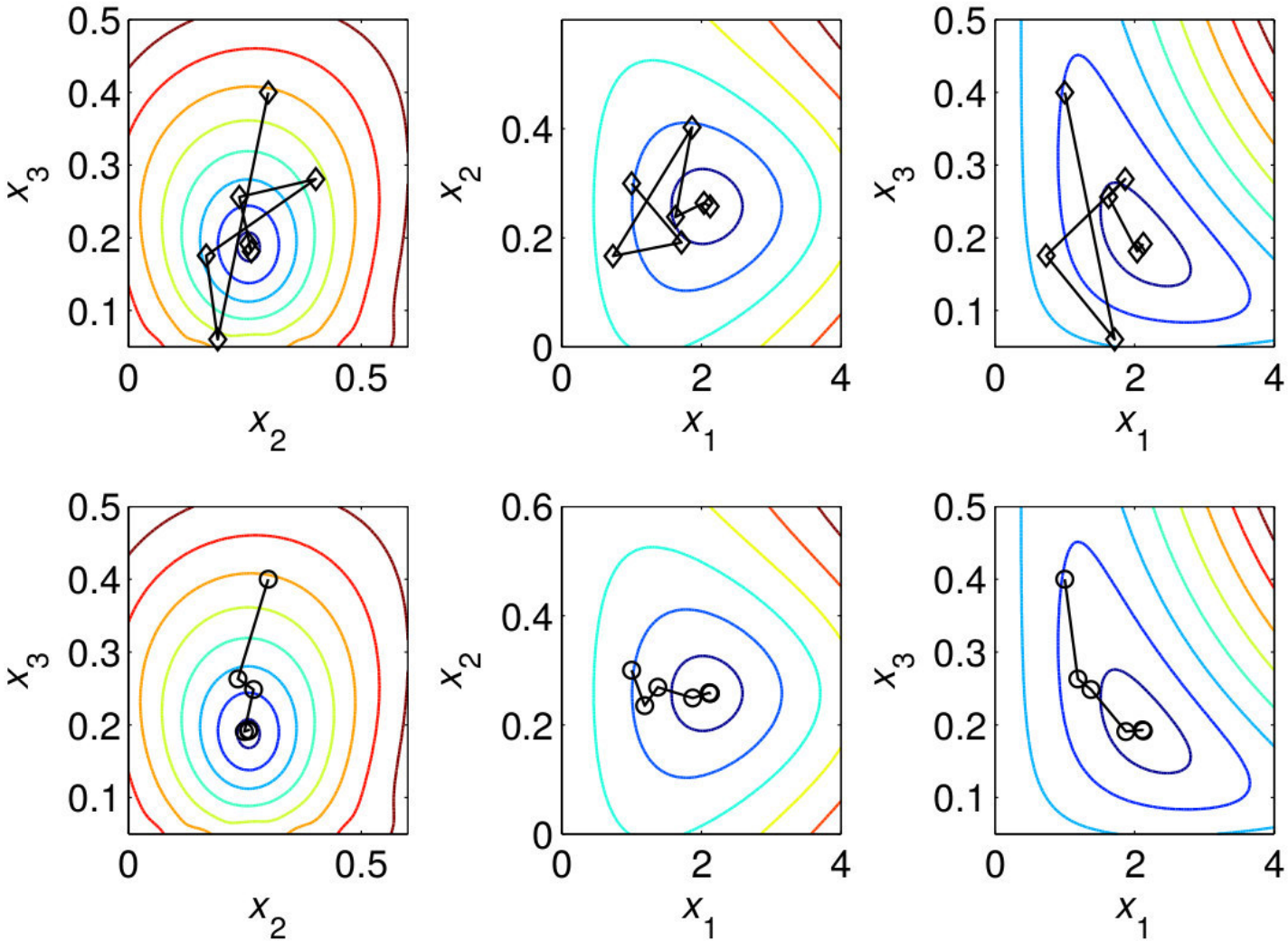
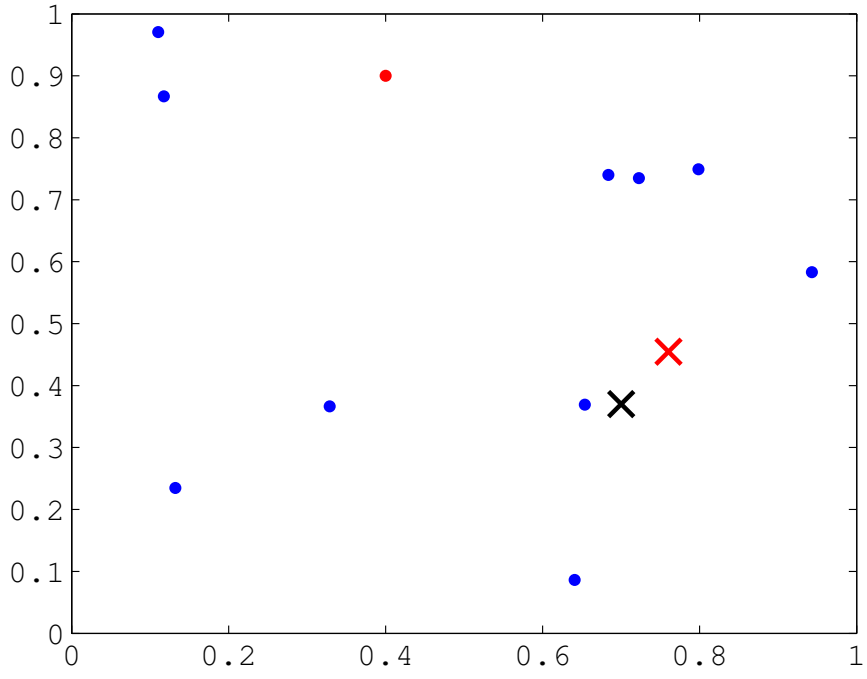


Figure: Gauss-Newton (top) and Levenberg-Marquardt (bottom) convergence

Nonlinear Least Squares: Example

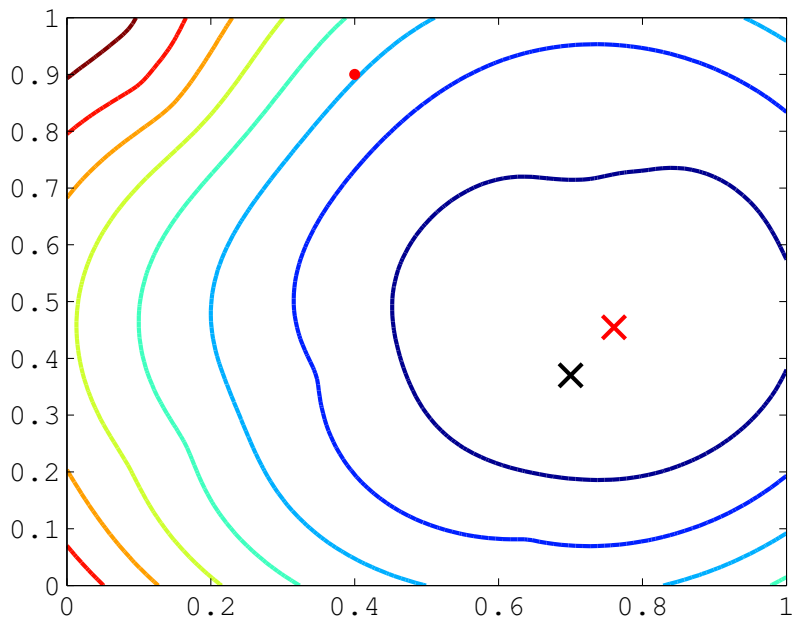
Python example: Using `lsqnonlin.py` we provide an initial guess (●), and converge to the solution (×)



Nonlinear Least Squares: Example

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

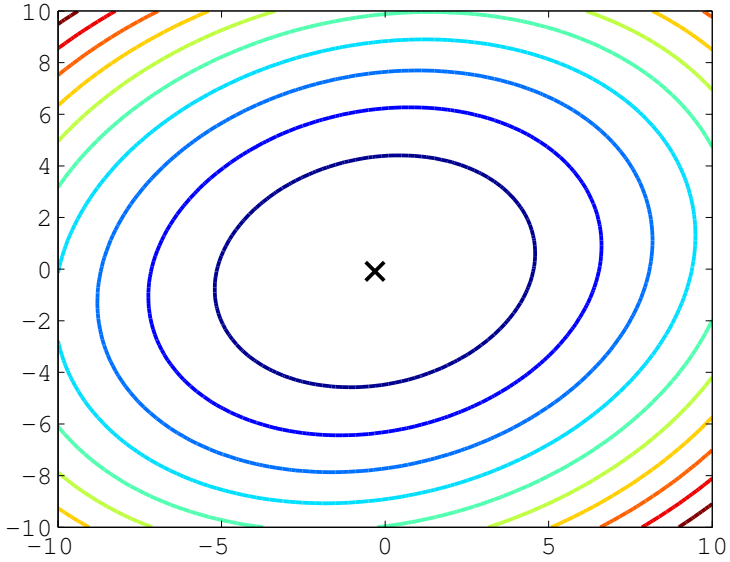
Recall $\times$ is the true transmitter location, $\times$ is our best-fit to the data; $\phi(\times) = 0.0248 < 0.0386 = \phi(\times)$.

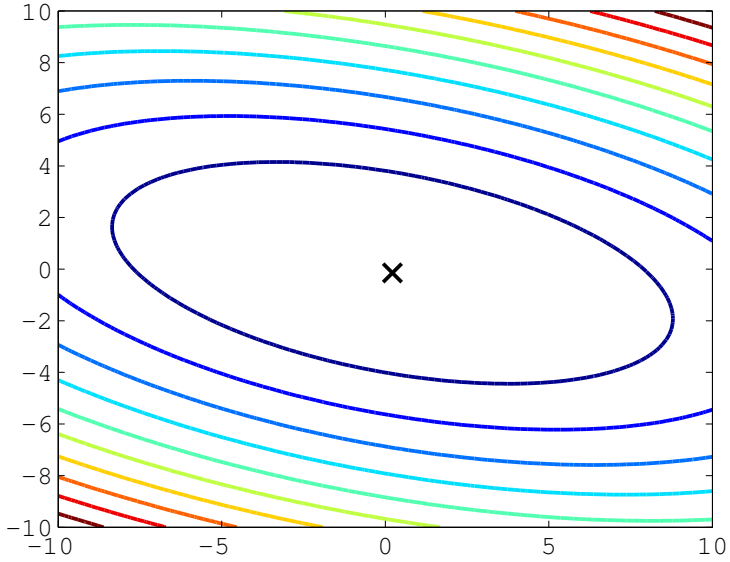


These contours are quite different from what we get in linear problems

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"