# Applied Math 205

- Homework 1 now posted. Due 5 PM on September 26.
- Last time: piecewise polynomial interpolation, least-squares fitting
- Today: least-squares, nonlinear least-squares

# The Normal Equations

For $A \in \mathbb{R}^{m \times n}$ with $m > n$, $A^T A$ is singular if and only if $A$ is rank-deficient.[1]

Proof:

($\Rightarrow$) Suppose $A^T A$ is singular. $\exists z \neq 0$ such that $A^T A z = 0$. Hence $z^T A^T A z = \|Az\|_2^2 = 0$, so that $Az = 0$. Therefore $A$ is rank-deficient.

($\Leftarrow$) Suppose $A$ is rank-deficient. $\exists z \neq 0$ such that $Az = 0$, hence $A^T A z = 0$, so that $A^T A$ is singular.

[1] Recall $A \in \mathbb{R}^{m \times n}$, $m > n$ is rank-deficient if columns are not L.I., i.e. $\exists z \neq 0$ s.t. $Az = 0$

# The Normal Equations

Hence if $A$ has full rank (i.e. $\text{rank}(A) = n$) we can solve the normal equations to find the unique minimizer $b$

However, in general it is a bad idea to solve the normal equations directly, it is not as numerically stable as some alternative methods

Question: If we shouldn't use normal equations, how do we actually solve least-squares problems ?

# Least-squares polynomial fit

Find least-squares fit for degree 11 polynomial to 50 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Let's express the best-fit polynomial using the monomial basis: $p(x; b) = \sum_{k=0}^{11} b_k x^k$

(Why not use the Lagrange basis? Lagrange loses its nice properties here since $m > n$, so we may as well use monomials)

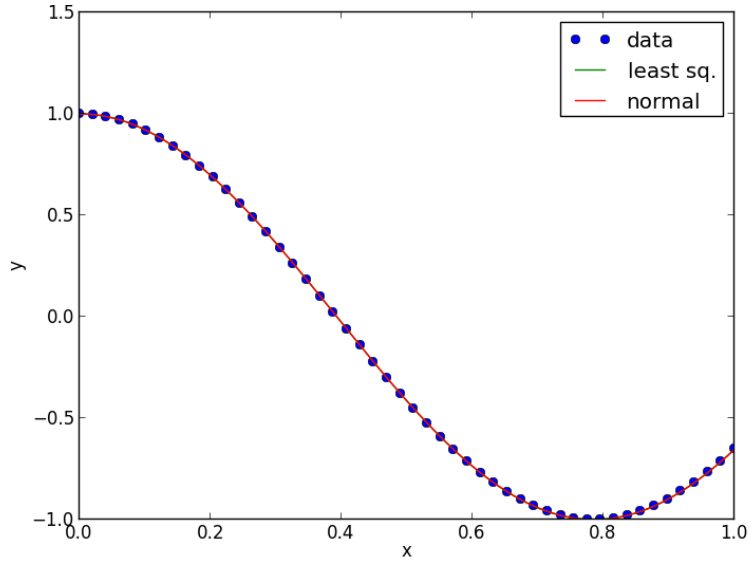
The $i$th condition we'd like to satisfy is $p(x_i; b) = \cos(4x_i) \implies$ over-determined system with "$50 \times 12$ Vandermonde matrix"

# Least-squares polynomial fit

But solving the normal equations still yields a small residual, hence we obtain a good fit to the data

$$\|r(b_{\text{normal}})\|_2 = \|y - Ab_{\text{normal}}\|_2 = 1.09 \times 10^{-8}$$

$$\|r(b_{\text{lst.sq.}})\|_2 = \|y - Ab_{\text{lst.sq.}}\|_2 = 8.00 \times 10^{-9}$$



# Non-polynomial Least-squares fitting

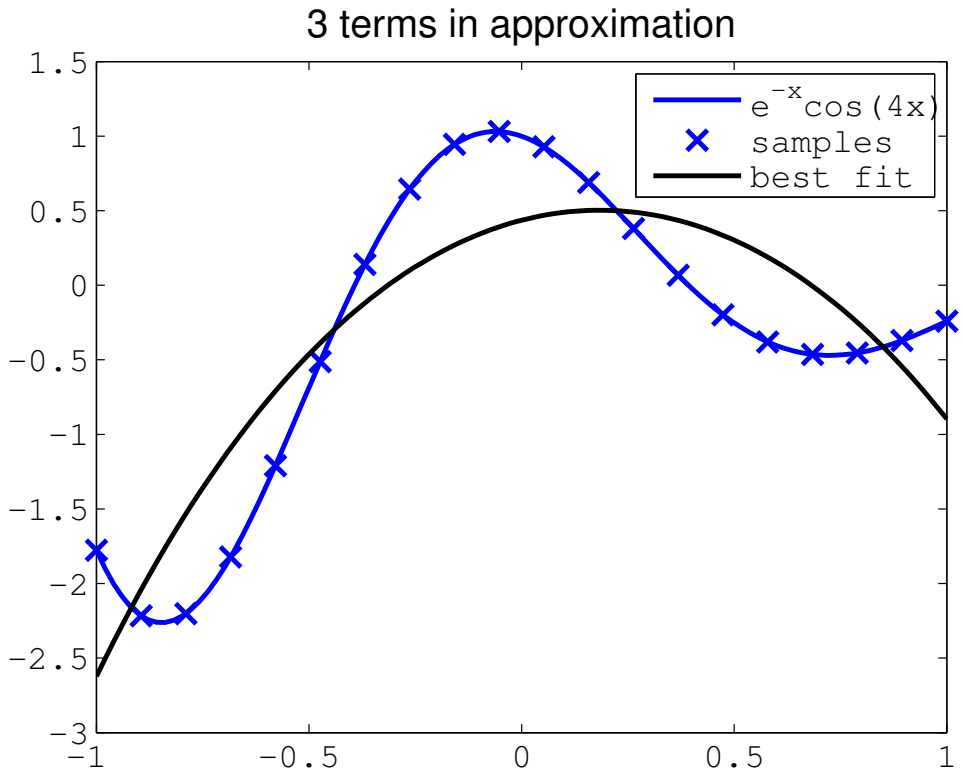
So far we have dealt with approximations based on polynomials, but we can also develop non-polynomial approximations

We just need the model to depend linearly on parameters

Example: Approximate $e^{-x}\cos(4x)$ using $f_n(x;b) \equiv \sum_{k=-n}^{n} b_k e^{kx}$

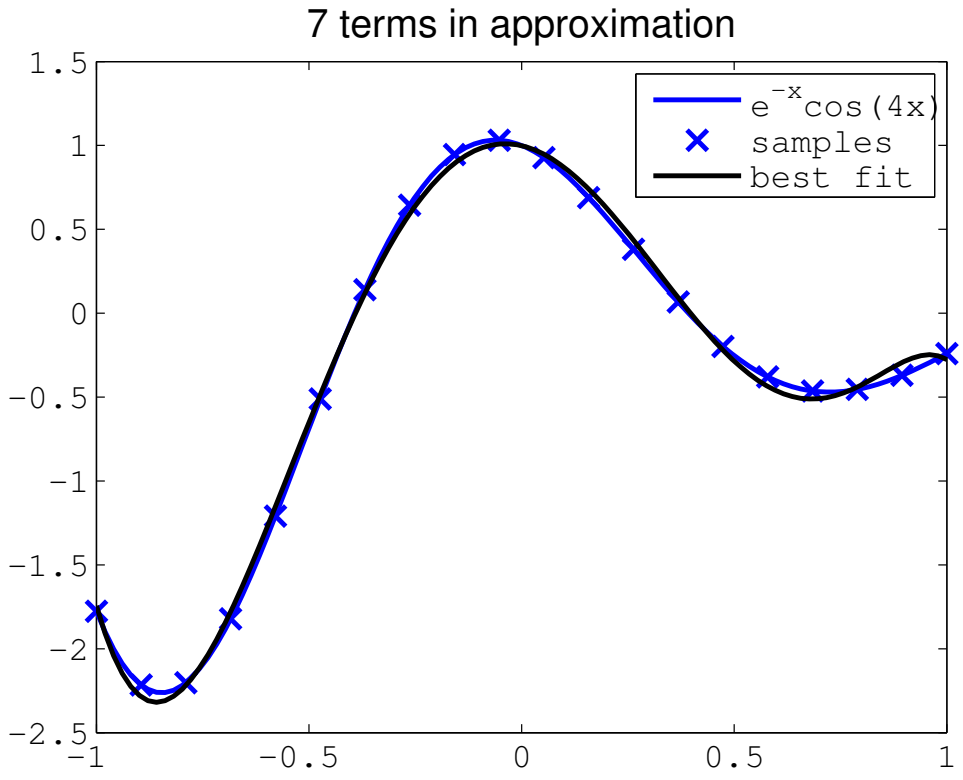
(Note that $f_n$ is linear in $b$: $f_n(x;\gamma a + \sigma b) = \gamma f_n(x;a) + \sigma f_n(x;b)$)

# Non-polynomial Least-squares fitting



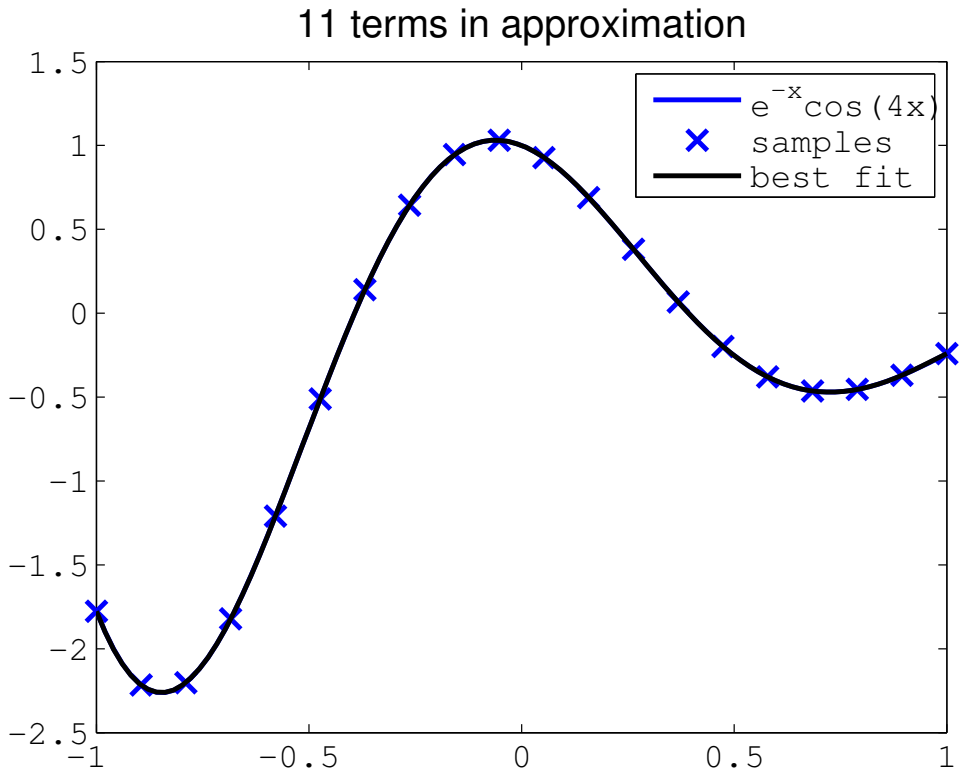

$$n = 1, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 4.16 \times 10^{-1}$$

# Non-polynomial Least-squares fitting




$$n = 3, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 1.44 \times 10^{-3}$$

# Non-polynomial Least-squares fitting




$$n = 5, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 7.46 \times 10^{-6}$$

# Pseudoinverse

Recall that from the normal equations we have:

$$A^T A b = A^T y$$

This motivates the idea of the "pseudoinverse" for $A \in \mathbb{R}^{m \times n}$:

$$A^+ \equiv (A^T A)^{-1} A^T \in \mathbb{R}^{n \times m}$$

Key point: $A^+$ generalizes $A^{-1}$, *i.e.* if $A \in \mathbb{R}^{n \times n}$ is invertible, then $A^+ = A^{-1}$

Proof: $A^+ = (A^T A)^{-1} A^T = A^{-1} (A^T)^{-1} A^T = A^{-1}$

# Pseudoinverse

Also:

- Even when $A$ is not invertible we still have still have $A^+A = I$
- In general $AA^+ \neq I$ (hence this is called a "left inverse")

And it follows from our definition that $b = A^+y$, i.e. $A^+ \in \mathbb{R}^{n \times m}$ gives the least-squares solution

Note that we define the pseudoinverse differently in different contexts

# Underdetermined Least Squares

So far we have focused on overconstrained systems (more constraints than parameters)

But least-squares also applies to underconstrained systems: $Ab = y$ with $A \in \mathbb{R}^{m \times n}$, $m < n$

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

i.e. we have a "short, wide" matrix $A$

# Underdetermined Least Squares

For $\phi(b) = \|r(b)\|_2^2 = \|y - Ab\|_2^2$, we can apply the same argument as before (i.e. set $\nabla\phi = 0$) to again obtain

$$A^T Ab = A^T y$$

But in this case $A^T A \in \mathbb{R}^{n\times n}$ has rank at most $m$ (where $m < n$), why?

Therefore $A^T A$ must be singular!

Typical case: There are infinitely vectors $b$ that give $r(b) = 0$, we want to be able to select one of them

# Underdetermined Least Squares

First idea, pose as a constrained optimization problem to find the feasible $b$ with minimum 2-norm:

$$\begin{array}{ll} \text{minimize} & b^T b \\ \text{subject to} & Ab = y \end{array}$$

This can be treated using Lagrange multipliers (discussed later in the Optimization section)

Idea is that the constraint restricts us to an $(n - m)$-dimensional hyperplane of $\mathbb{R}^n$ on which $b^T b$ has a unique minimum

# Underdetermined Least Squares

We will show later that the Lagrange multiplier approach for the above problem gives:

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

As a result, in the underdetermined case the pseudoinverse is defined as $A^+ = A^T(AA^T)^{-1} \in \mathbb{R}^{n\times m}$

Note that now $AA^+ = I$, but $A^+A \neq I$ in general (*i.e.* this is a "right inverse")

# Underdetermined Least Squares

Here we consider an alternative approach for solving the underconstrained case

Let's modify $\phi$ so that there is a unique minimum!

For example, let

$$\phi(b) \equiv \|r(b)\|_2^2 + \|Sb\|_2^2$$

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

This is called regularization: we make the problem well-posed ("more regular") by modifying the objective function

# Underdetermined Least Squares

Calculating $\nabla\phi = 0$ in the same way as before leads to the system

$$(A^T A + S^T S)b = A^T y$$

We need to choose $S$ in some way to ensure $(A^T A + S^T S)$ is invertible

Can be proved that if $S^T S$ is positive definite then $(A^T A + S^T S)$ is invertible

Simplest positive definite regularizer: $S = \mu \mathrm{I} \in \mathbb{R}^{n \times n}$ for $\mu \in \mathbb{R}_{>0}$

# Underdetermined Least Squares

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

12 parameters, 5 constraints $\implies$ $A \in \mathbb{R}^{5\times 12}$

We express the polynomial using the monomial basis (can't use Lagrange since $m \neq n$): $A$ is a submatrix of a Vandermonde matrix

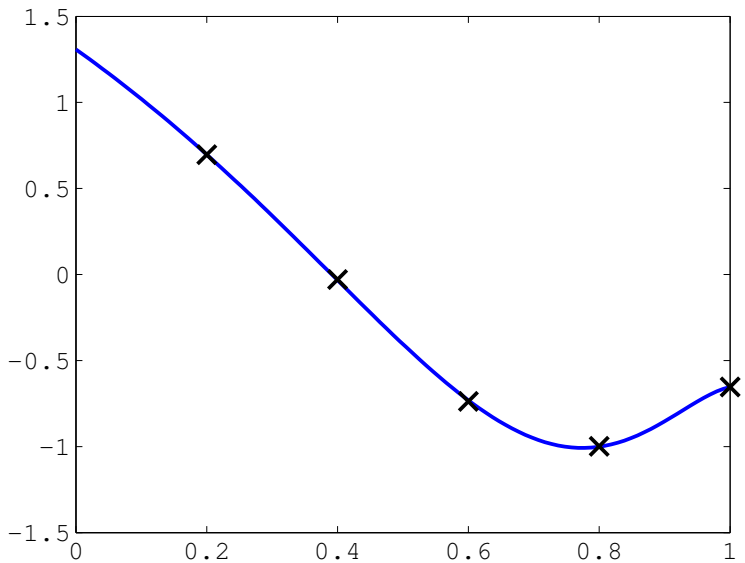
If we naively use the normal equations we see that $\text{cond}(A^T A) = 4.78 \times 10^{17}$, *i.e.* "singular to machine precision"!

Let's see what happens when we regularize the problem with some different choices of $S$

# 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. $\mu = 0.001$)




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

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

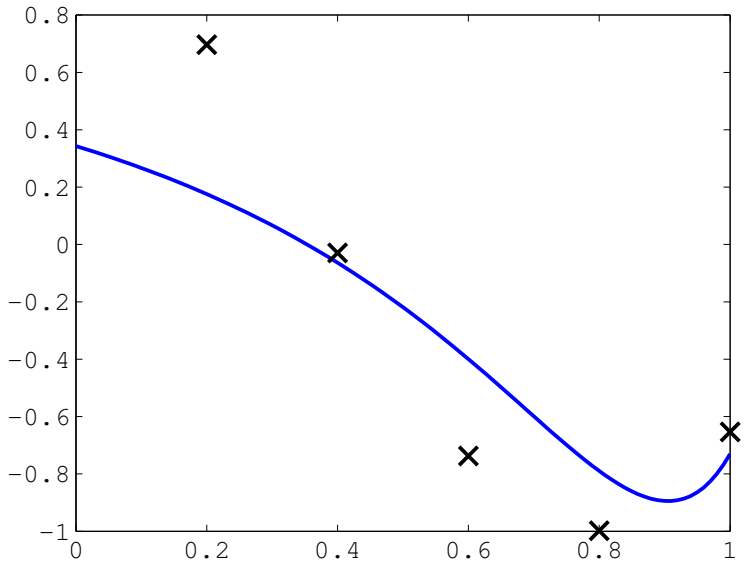
$$\mathrm{cond}(A^T A + S^T S) = 1.54 \times 10^7$$

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. $\mu = 0.5$)




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

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

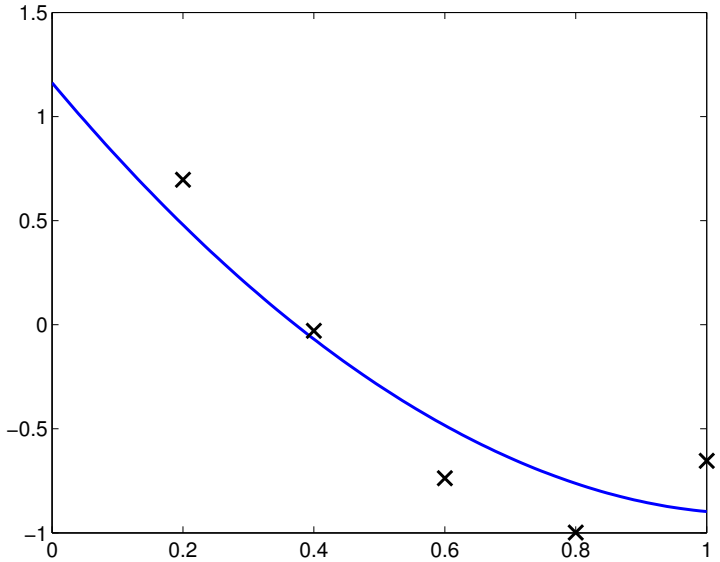
$$\mathrm{cond}(A^T A + S^T S) = 62.3$$

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




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

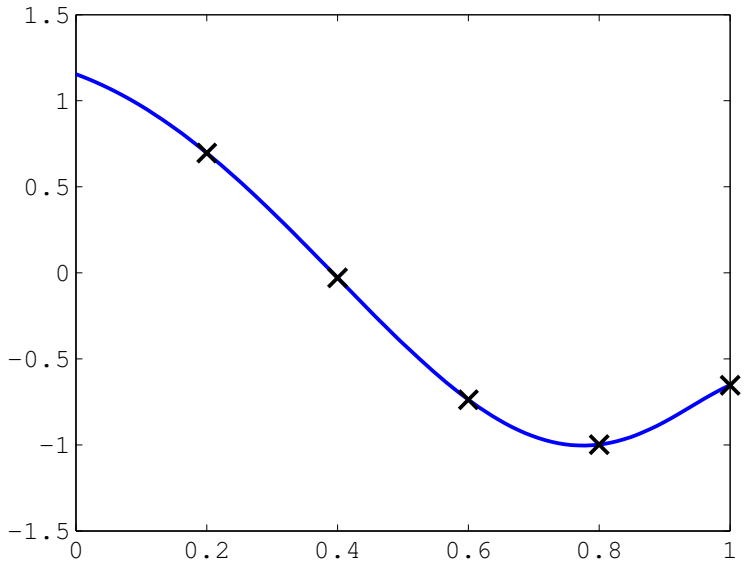
$$\|b\|_2 = 4.27$$

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

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

So far we have looked at finding a "best fit" solution to a linear system (linear least-squares)

A more difficult situation is when we consider least-squares for nonlinear systems

Key point: We are referring to linearity in the parameters, not linearity of the model

(e.g. polynomial $p_n(x; b) = b_0 + b_1 x + \ldots + b_n x^n$ is nonlinear in $x$, but linear in $b$!)

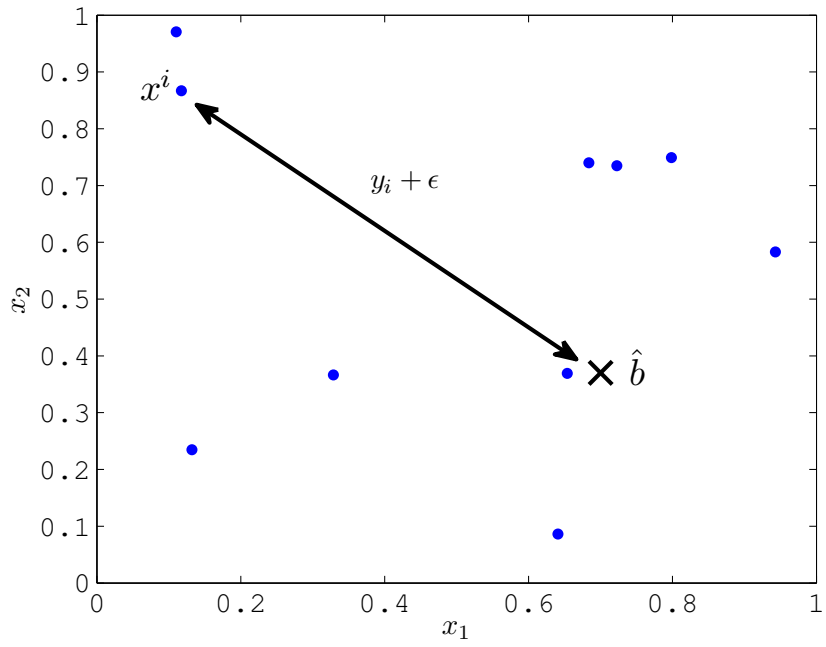
In nonlinear least-squares, we fit functions that are nonlinear in the parameters

# 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 ($\epsilon$)



# 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 in the linear case, we seek to minimize $\phi$ by finding $b$ such that $\nabla\phi = 0$

We have $\phi(b) = \frac{1}{2}\sum_{j=1}^{m}[r_j(b)]^2$

Hence for the $i^{th}$ component of the gradient vector, we have

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

# Nonlinear Least Squares

This is equivalent to $\nabla\phi = J_r(b)^T r(b)$ where $J_r(b) \in \mathbb{R}^{m\times n}$ is the Jacobian matrix of the residual

$$\{J_r(b)\}_{ij} = \frac{\partial r_i(b)}{\partial b_j}$$

Exercise: Show that $J_r(b)^T r(b) = 0$ reduces to the normal equations when the residual is linear

# Nonlinear Least Squares

Hence we seek $b \in \mathbb{R}^n$ such that:

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

This has $n$ equations, $n$ unknowns; in general this is a nonlinear system that we have to solve iteratively

An important recurring theme is that linear systems can be solved in "one shot," whereas nonlinear generally requires iteration

We will briefly introduce Newton's method for solving this system and defer detailed discussion until the optimization section

# Nonlinear Least Squares

Recall Newton's method for a function of one variable: find $x \in \mathbb{R}$ such that $f(x) = 0$

Let $x_k$ be our current guess, and $x_k + \Delta x = x$, then Taylor expansion gives

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

It follows that $f'(x_k)\Delta x \approx -f(x_k)$ (approx. since we neglect the higher order terms)

This motivates Newton's method: $f'(x_k)\Delta x_k = -f(x_k)$, where $x_{k+1} = x_k + \Delta x_k$

# Nonlinear Least Squares

This argument generalizes directly to functions of several variables

For example, suppose $F : \mathbb{R}^n \to \mathbb{R}^n$, then find $x$ s.t. $F(x) = 0$ by

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

where $J_F$ is the Jacobian of $F$, $\Delta x_k \in \mathbb{R}^n$, $x_{k+1} = x_k + \Delta x_k$

# Nonlinear Least Squares

In the case of nonlinear least squares, to find a stationary point of $\phi$ we need to find $b$ such that $J_r(b)^T r(b) = 0$

That is, we want to solve $F(b) = 0$ for $F(b) \equiv J_r(b)^T r(b)$

We apply Newton's Method, hence need to find the Jacobian, $J_F$, of the function $F : \mathbb{R}^n \to \mathbb{R}^n$

# Nonlinear Least Squares

Consider $\frac{\partial F_i}{\partial b_j}$ (this will be the $ij$ entry of $J_F$):

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

# Gauss–Newton Method

It is generally a pain to deal with the second derivatives in the previous formula, second derivatives get messy!

Key observation: As we approach a good fit to the data, the residual values $r_k(b)$, $1 \leq k \leq m$, should be small

Hence we omit the term $\sum_{k=1}^{m} r_k \frac{\partial^2 r_k}{\partial b_i \partial b_j}$.

# Gauss–Newton Method

Note that $\sum_{k=1}^{m} \frac{\partial r_k}{\partial b_j} \frac{\partial r_k}{\partial b_i} = (J_r(b)^T J_r(b))_{ij}$, so that when the residual is small $J_F(b) \approx J_r(b)^T J_r(b)$

Then putting all the pieces together, we obtain the iteration: $b_{k+1} = b_k + \Delta b_k$ where

$$J_r(b_k)^T J_r(b_k) \Delta b_k = -J(b_k)^T r(b_k), \qquad k = 1, 2, 3, \ldots$$

This is known as the Gauss–Newton Algorithm for nonlinear least squares