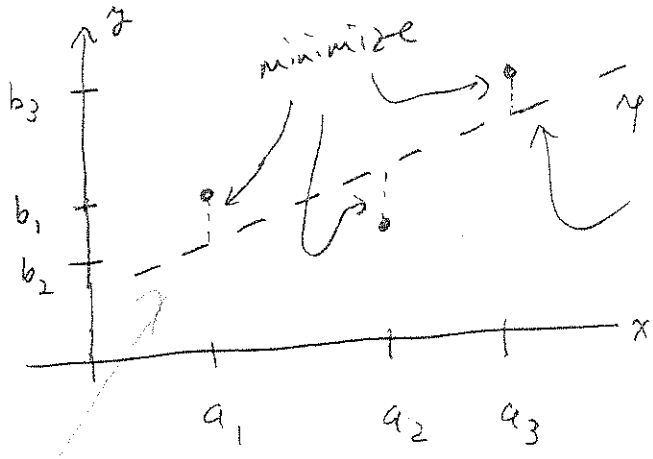
Least Squares Fitting

Example: Fit straight line $y = cx + d$

to data (a_1, b_1), (a_2, b_2), (a_3, b_3)



We can't expect straight line to interpolate more than 2 data points at once, even though we are using a "linear model" of the data.

i.e. we would like to find c, d s.t.

$$\left.\begin{aligned} d + c a_1 &= b_1 \\ d + c a_2 &= b_2 \\ d + c a_3 &= b_3 \end{aligned}\right\}$$

3 eqs. in 2 unknowns c, d

↖ Cannot expect to solve so minimize distances to data points

or in matrix form

$$Ax = \begin{bmatrix} 1 & a_1 \\ 1 & a_2 \\ 1 & a_3 \end{bmatrix} \begin{bmatrix} d \\ c \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = b$$

A: given 3×2 matrix "input"

x: unknown model parameters, 2×1 vector

b: given 3×1 vector "output"

$\therefore$ we will try to find c, d s.t.

$$\|Ax-b\|_2^2 = \sum_{i=1}^{3} (ca_i + d - b_i)^2 = \begin{cases} \text{sum of squares} \\ \text{of distances} \\ \text{of } y = cx+d \\ \text{to } (a_i, b_i)\text{'s.} \end{cases}$$

is minimized, $(x_1 = d,\ x_2 = c)$

i.e. Solve $\min_{x \in \mathbb{R}^2} \|Ax-b\|_2$ = linear least squares problem (linear regression)

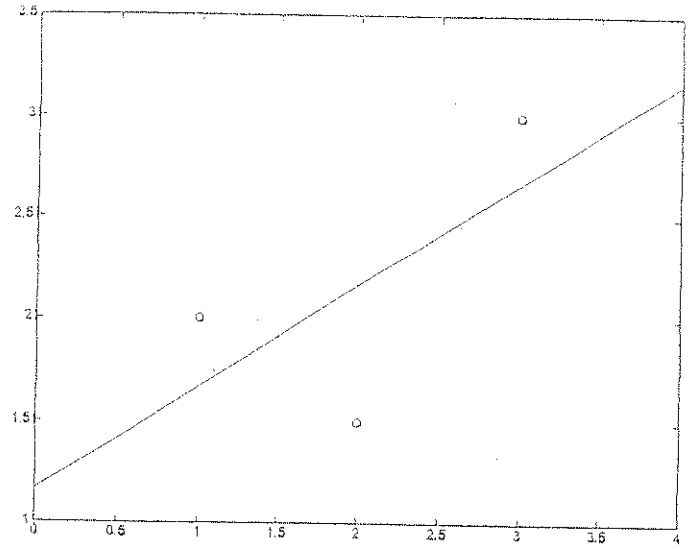
(Other norms like $\|\cdot\|_1$ or $\|\cdot\|_\infty$ could be used, but $\|\cdot\|_2$ leads to nice algorithms since orthogonal matrices preserve $\|\cdot\|_2$. Also, $\|Ax-b\|_2$ has a natural statistical meaning, the standard deviation.)

More generally, the linear least squares problem is

$$(LS) \begin{cases} \text{Given } A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^{m \times 1} \\ \text{find } x \in \mathbb{R}^{n \times 1} \text{ s.t. } \|Ax-b\|_2 \text{ is} \\ \text{minimized} \end{cases}$$

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix},\ x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix},\ b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

$$\|Ax-b\|_2 = \sqrt{\sum_{i=1}^{m} (A(i,:)x - b(i))^2} = \sqrt{\sum_{i=1}^{m} \left(\left(\sum_{j=1}^{n} a_{ij} x_j \right) - b_i \right)^2},$$



An example of a least squares fit.

```
» A=[1 1;1 2;1 3]

A =

     1     1
     1     2
     1     3

» b=[2;1.5;3]

b =

   2.00000000000000
   1.50000000000000
   3.00000000000000

» x=A\b

x =

   1.16666666666667
   0.50000000000000

» x1=linspace(0,4,100);
» y1=x(1)+x(2)*x1;
» plot(x1,y1,A(:,2),b,'o')
»
```

```
» delta=.0001

delta =

    1.000000000000000e-004

» A=[1 1;1 1-delta;1 1-2*delta]

A =

   1.00000000000000   1.00000000000000
   1.00000000000000   0.99990000000000
   1.00000000000000   0.99980000000000

» b=[2;2-delta;2-2*delta]

b =

   2.00000000000000
   1.99990000000000
   1.99980000000000

» x=A\b

x =

   1.00000000000096
   0.99999999999904

» x=(A'*A)\(A'*b)

x =

   1.00000006660672
   0.99999993338662

» cond(A)

ans =

    2.449244810141778e+004

» cond(A'*A)

ans =

    5.998799888590298e+008

»
```

← Solve least squares $\min \|Ax-b\|_2$

error

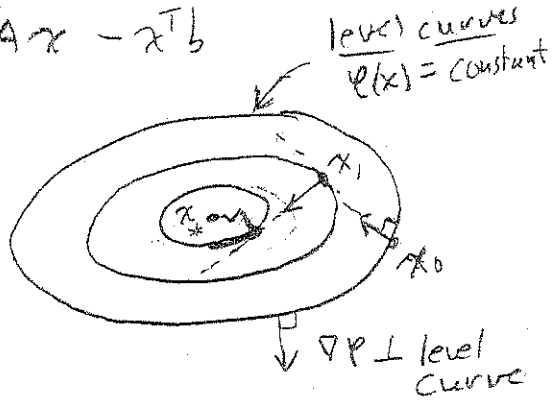
error

← Solve normal eqs. $A^TAx = A^Tb$

Math 551 DeTurck

Steepest descent - an easy example of an iterative method for solving $Ax = b$, A sym. pos. def.

Quadratic form $\varphi(x) = \frac{1}{2} x^T A x - x^T b$



$\nabla\varphi = Ax - b = -r$

$r = b - Ax$ = residual

minimum of φ occurs where $\nabla\varphi = Ax - b = 0$

i.e. at $x = x_* =$ exact solution

Steepest descent: initial guess x_0

search direction $r_0 = -\nabla\varphi(x_0) = b - Ax_0$

minimize $\varphi(x)$ in direction r_0 from x_0

$$\frac{d}{d\alpha}\varphi(x_0 + \alpha r_0) = \frac{d}{d\alpha}\left(\frac{1}{2}(x_0+\alpha r_0)^T A(x_0+\alpha r_0) - (x_0+\alpha r_0)^T b\right)$$

$$= \frac{d}{d\alpha}\left(\frac{1}{2}x_0^T A x_0 + \alpha r_0^T A x_0 + \frac{1}{2}\alpha^2 r_0^T A r_0 - x_0^T b - \alpha r_0^T b\right)$$

$$= r_0^T A x_0 + \alpha r_0^T A r_0 - r_0^T b$$

$$= \alpha r_0^T A r_0 - r_0^T (b - Ax_0)$$

$$= \alpha r_0^T A r_0 - r_0^T r_0$$

$$= 0 \implies \alpha = \alpha_1 = r_0^T r_0 / r_0^T A r_0$$

Then $x_1 = x_0 + \alpha_1 r_0$

and the iteration is repeated.

Steepest descent iteration

$x_0 = 0$ (standard initial guess)

$r_0 = b$

for $n = 1, 2, 3, \ldots$

$\alpha_n = r_{n-1}^T r_{n-1} / r_{n-1}^T A r_{n-1}$ step length

$x_n = x_{n-1} + \alpha_n r_{n-1}$ approximate soln.

$r_n = r_{n-1} - \alpha_n A r_{n-1}$ residual (= search direction)

Recall $r_n = b - Ax_n$ ← leads to cancellation as $x_n \to x_*$

$= b - A(x_{n-1} + \alpha_n r_{n-1})$

$= b - Ax_{n-1} - \alpha_n A r_{n-1}$

$= r_{n-1} - \alpha_n A r_{n-1}$ ← better numerically

Compare with conjugate gradient which minimizes $\|e_n\|_A^2$ over K_n. But x_* is unknown, in general, and so is $e_n = x_* - x_n$.

But note

$\|e_n\|_A^2 = e_n^T A e_n = (x_* - x_n)^T A (x_* - x_n)$

$= x_n^T A x_n - 2 x_n^T A x_* + x_*^T A x_*$ $(Ax_* = b)$

$= x_n^T A x_n - 2 x_n^T b + x_*^T b$

$= 2\phi(x_n) + \text{constant}$

So CG minimizes same ϕ as steepest descent.