Lecture 16: Least squares

Admin:

Natural problems for solving linear systems:

1. Solve $Ax = b$ for x
2. Assuming $Ax = b$ has a solution, find the shortest solution x. (Today)
3. "Least squares": If $Ax = b$ has no solution ($b \notin R(A)$), find an x that minimizes $\|Ax - b\|$.
4. "Compressed sensing": Find the sparsest x such that $Ax = b$.

Simple application of SVD:

Theorem: For any matrix A,

- $\text{rank}(A^TA) = \text{rank}(A)$
- $R(A^TA) = R(A^T)$
- $N(A^TA) = N(A)$.

Proof: We have seen this already, but it is much easier to prove using the SVD.

Let the SVD of A be

$$A = \sum_i \lambda_i \vec{u}_i \vec{v}_i^T \quad \Rightarrow \text{rank}(A) = \#\{i \mid \lambda_i > 0\}$$

$$\Rightarrow A^TA = \left(\sum_i \lambda_i v_i u_i^T\right)\left(\sum_j \lambda_j u_j v_j^T\right)$$

$$= \sum_i \lambda_i^2 v_i v_i^T \quad \text{since } u_i \cdot u_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

This is a* SVD for A^TA $\Rightarrow \text{rank}(A^TA) = \#\{i \mid \lambda_i^2 > 0\}$

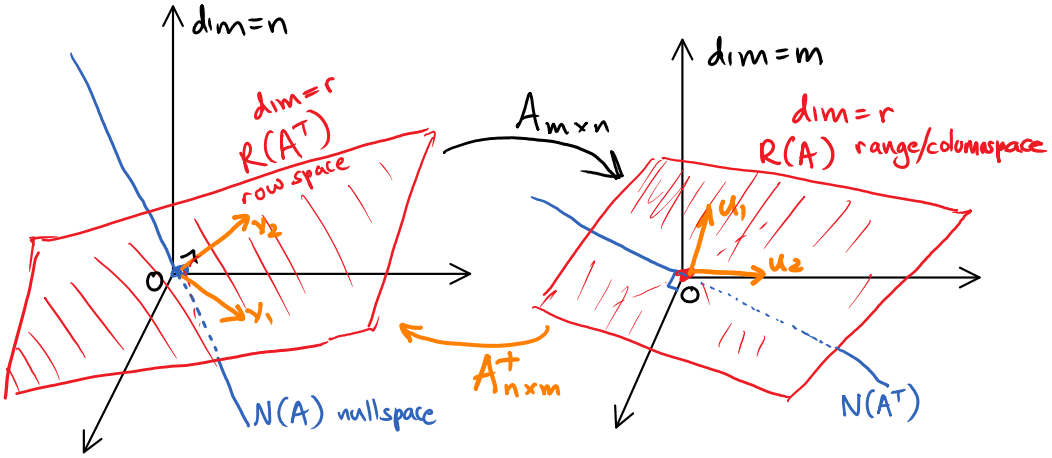
$\Rightarrow \text{rank}(A^TA) = \text{rank}(A)$. ✓

The other statements are all similar (exercises). □

Definition: Matrix pseudoinverse

↑ dim = n

↑ dim = m



The **pseudoinverse** of A, denoted A^+, is the unique linear operator $\mathbb{R}^m \to \mathbb{R}^n$ with

- $N(A^+) = R(A)^\perp = N(A^T)$
- $R(A^+) = R(A^T) = N(A)^\perp$
- $A^+A = P_{R(A^T)}$

↖ plus sign; not to be confused with adjoint ($\dagger$) or transpose (T)

If the SVD of A is

$$A = \sum_i \lambda_i \vec{u}_i \vec{v}_i^T,$$

then A^+ is given by

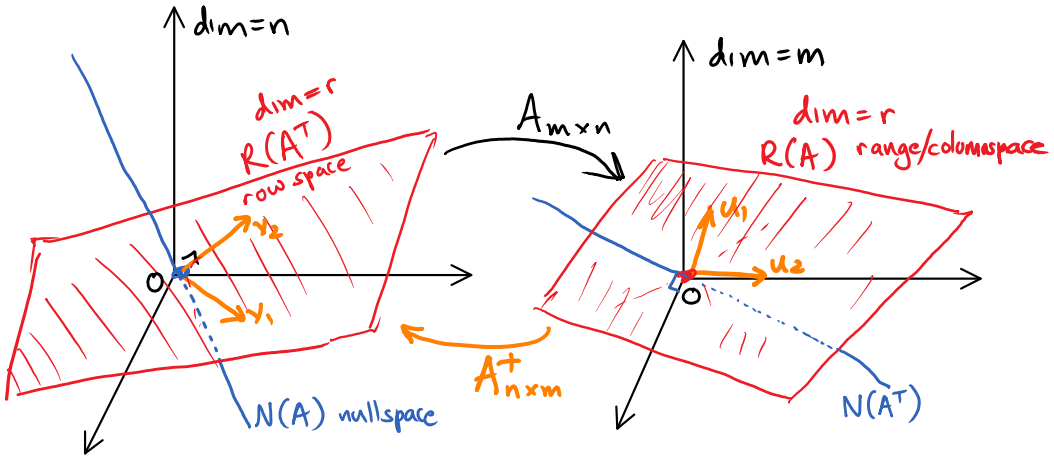
$$\boxed{A^+ = \sum_{i:\lambda_i>0} \frac{1}{\lambda_i} \vec{v}_i \vec{u}_i^T}$$

Properties of the (Moore-Penrose) pseudoinverse:

$$A = \sum_i \lambda_i \vec{u}_i \vec{v}_i^T \qquad A^+ = \sum_{i:\lambda_i>0} \frac{1}{\lambda_i} \vec{v}_i \vec{u}_i^T$$

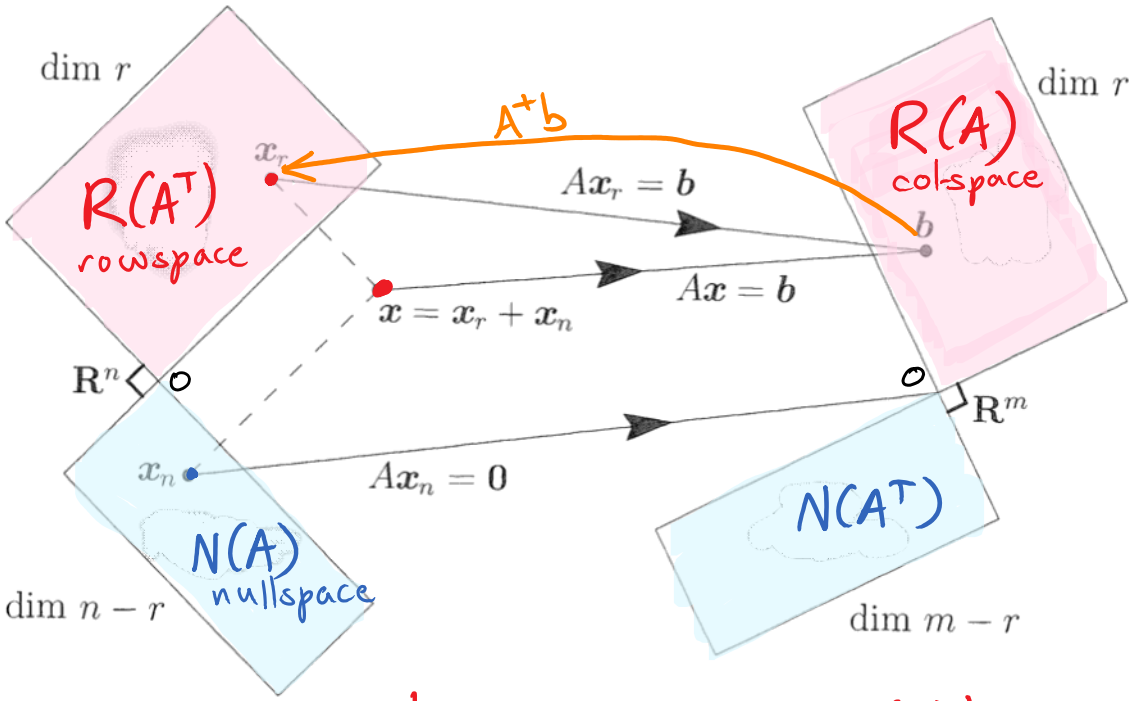
- $AA^+ = \sum_{i:\lambda_i>0} \vec{u}_i \vec{u}_i^T$ = Projection onto $R(A)$
- $A^+A = \sum_{i:\lambda_i>0} \vec{v}_i \vec{v}_i^T$ = Projection onto $R(A^T)$

$\Rightarrow$ (for example) $AA^+A = AP_{R(A)} = P_{R(A^T)}A = A$



- If A is invertible/nonsingular, then $A^+ = A^{-1}$.

Corollary: If $Ax = b$ has multiple solutions, ie., $N(A) \neq \{\vec{0}\}$, then $x = A^+b$ is the shortest solution.



$N(A) = R(A^T)^{\perp}$ $\quad$ $N(A^T) = R(A)^{\perp}$

Why?

Any solution $x \in A^+b + N(A)$

A^+b: particular solution

$N(A)$: set of solutions to the homogeneous equations $Ax = 0$

Since $A^+b \in R(A^T)$, which is $\perp$ to $N(A)$,

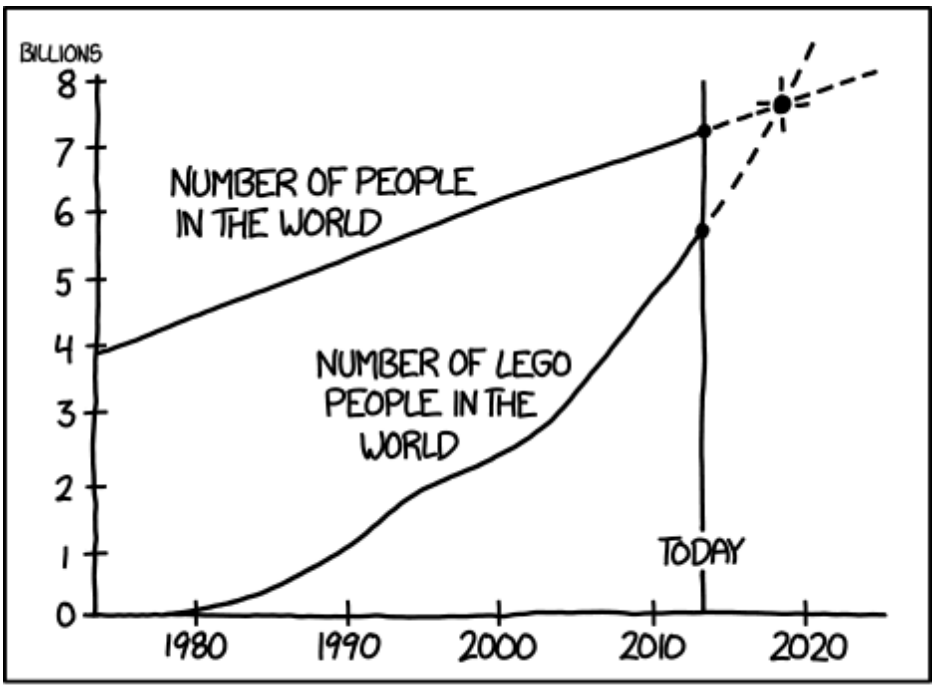
$\|x\|^2 = \|A^+b\|^2 + \|\text{its component in } N(A)\|^2$

which is minimal if the component in $N(A)$ is 0,

ie., $x = A^+b$. ✓ □

"Least Squares" Fitting & Applications to Data Analysis

Reading:
Meyer 4.6
5.13-14
Strang 3.3



BY 2019, HUMANS WILL BE OUTNUMBERED.

[xkcd.com/128]

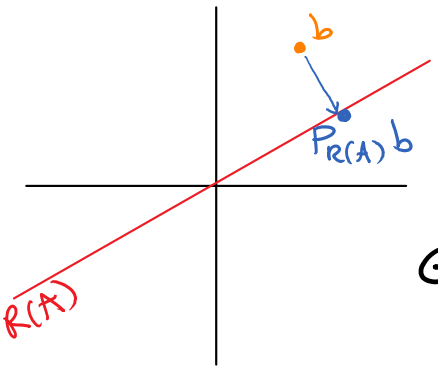
Typical problem: System of equations

$$A\vec{x} = \vec{b}$$

but $\vec{b} \notin R(A)$!

⇒ there is no solution $\vec{x}$

⇒ Instead, find x to minimize $\|Ax-b\|$



Geometrically:

Project b into $R(A)$, solve

$$Ax = P_{R(A)}b$$

Example:

$$\underbrace{\begin{pmatrix} 1 & -1 \\ 2 & 3 \\ 0 & 0 \end{pmatrix}}_{A}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 \\ 5 \\ 10 \end{pmatrix}}_{b}$$

There is no solution, since $(Ax)_3 = 0$ always.

But $R(A) = \text{Span}\left\{\begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 3 \\ 0 \end{pmatrix}\right\}$

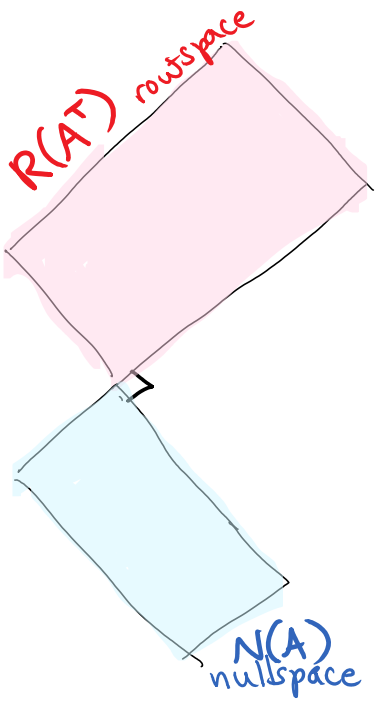
$= xy$-plane in $\mathbb{R}^3$

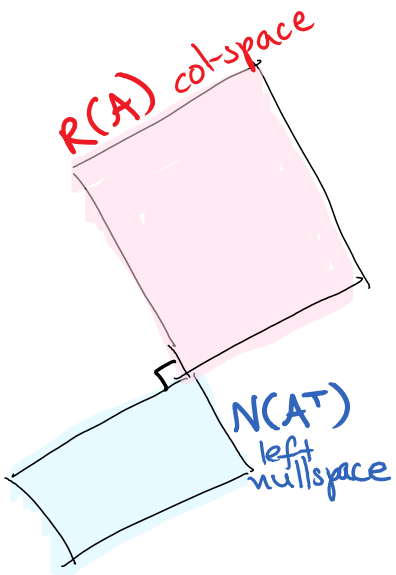
$\Rightarrow$ min-error solution solves

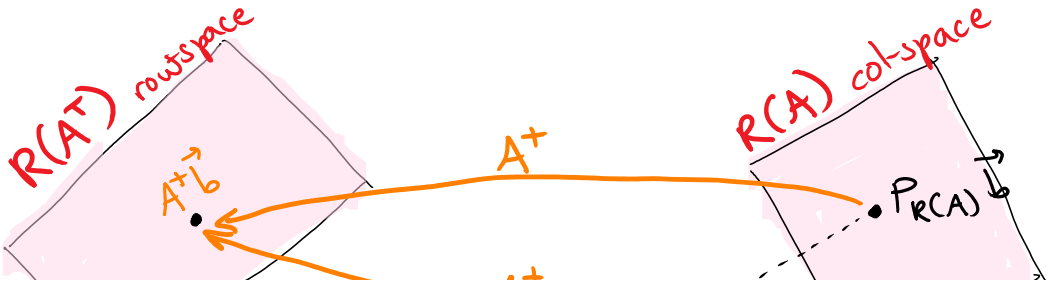
$$A\vec{x} = P_{R(A)}\vec{b} = \begin{pmatrix} 0 \\ 5 \\ 0 \end{pmatrix}$$

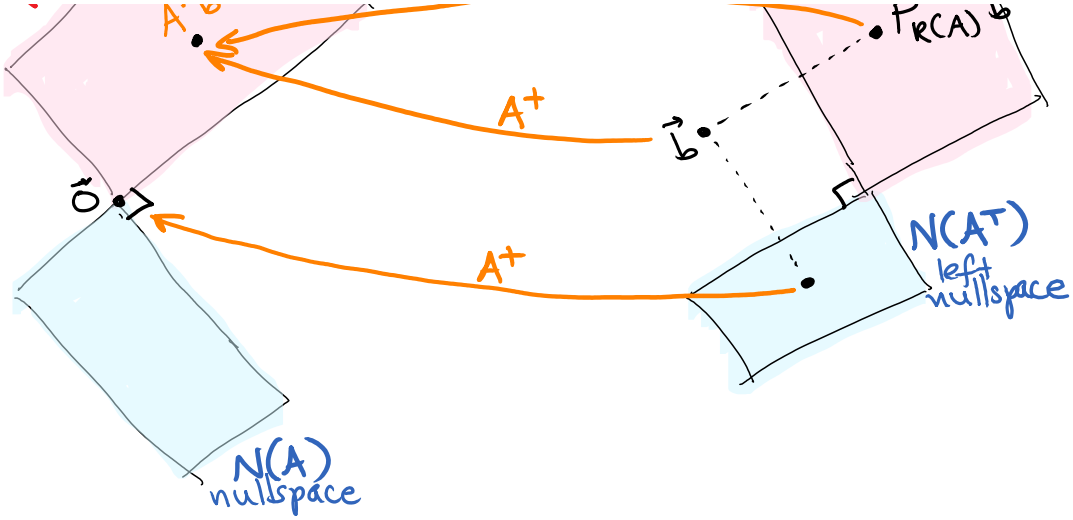
$$\Rightarrow \vec{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \checkmark$$

Another approach...









$$\Rightarrow x = A^+ b$$
minimizes $\|Ax - b\|$

Algebraically:

$$A = \sum_i \sigma_i \vec{u}_i \vec{v}_i \quad \text{(SVD)}$$

$$R(A) = \text{Span}(\{\vec{u}_i \mid \sigma_i > 0\})$$

$$P_{R(A)} = \sum_{i:\sigma_i>0} \vec{u}_i \vec{u}_i^T \quad \text{(since they are orthonormal)}$$

We want to solve

$$A\vec{x} = P_{R(A)}\vec{b}$$

$$\sum_{i:\sigma_i>0} \sigma_i (v_i \cdot x)\vec{u}_i \qquad \sum_{i:\sigma_i>0} (u_i \cdot b)\vec{u}_i$$

$$\Rightarrow v_i \cdot x = \begin{cases} \frac{1}{\sigma_i}(u_i \cdot b) & \text{if } \sigma_i > 0 \\ 0 & \text{if } \sigma_i = 0 \end{cases}$$

$$\Rightarrow \vec{x} = \sum_i (v_i \cdot x)\vec{v}_i$$

$$= \sum_{i:\sigma_i>0} \frac{1}{\sigma_i} \vec{v}_i (u_i \cdot b)$$

$$= A^+ \vec{b}$$

$\Rightarrow x = A^+ b$
minimizes $\|Ax - b\|$

Example:

```
>> A = [1 -1; 2 3; 0 0]

A =

     1    -1
     2     3
     0     0

>> format long e
>> [U,D,V] = svd(A)
```

left singular vectors

```
U =

  -8.980559531591714e-02   9.959593139531120e-01                     0
   9.959593139531121e-01   8.980559531591698e-02                     0
                         0                         0   1.000000000000000e+00


D =

   3.618033988749895e+00                         0
                         0   1.381966011250106e+00
                         0                         0
```

singular values

right singular vectors

```
V =

   5.257311121191336e-01   8.506508083520399e-01
   8.506508083520399e-01  -5.257311121191336e-01

>> Apseudoinv = V * [1/3.618033988749895 0 0; 0 1/1.381966011250106 0] * U'

Apseudoinv =

    5.999999999999995e-01   1.999999999999999e-01                     0
   -3.999999999999997e-01   2.000000000000000e-01                     0

>> Apseudoinv * [0; 5; 10]

ans =

   9.999999999999996e-01
   1.000000000000000e+00
```

$= \begin{pmatrix} .6 & .2 & 0 \\ -.4 & .2 & 0 \end{pmatrix}$

$= \frac{1}{5}\begin{pmatrix} 3 & 1 & 0 \\ -2 & 1 & 0 \end{pmatrix}$

$= \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ ✓

```
>> format short e
```

```
>> format short e
>> Apseudoinv2 = pinv(A)

Apseudoinv2 =

   6.0000e-01   2.0000e-01            0
  -4.0000e-01   2.0000e-01            0
```

Matlab's built-in pseudo-inverse function

← same as we got above!

```
>> A * Apseudoinv

ans =

   1.0000e+00  -1.3878e-16            0
            0   1.0000e+00            0
            0            0            0
```

Observe:

AA^+ = projection onto xy-plane

```
>> Apseudoinv * A

ans =

   1.0000e+00   1.1102e-16
   3.8858e-16   1.0000e+00
```

$A^+A = I$ on $\mathbb{R}^2$

APPLICATION : LINEAR REGRESSION

Example: Find the equation for the line $y = a + bx$ that goes through the points $(2, 5)$ and $(4, 11)$.

Answer: $a + 2 \cdot b = 5$, $a + 4 \cdot b = 11$ $\Leftrightarrow \begin{pmatrix} 1 & 2 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 5 \\ 11 \end{pmatrix}$

```
>> [1 2; 1 4] \ [5; 11]

ans =

   -1
    3
```

$\Rightarrow y = -1 + 3x$

Observe: $\begin{pmatrix} 1 & 2 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 5 \\ 11 \end{pmatrix}$

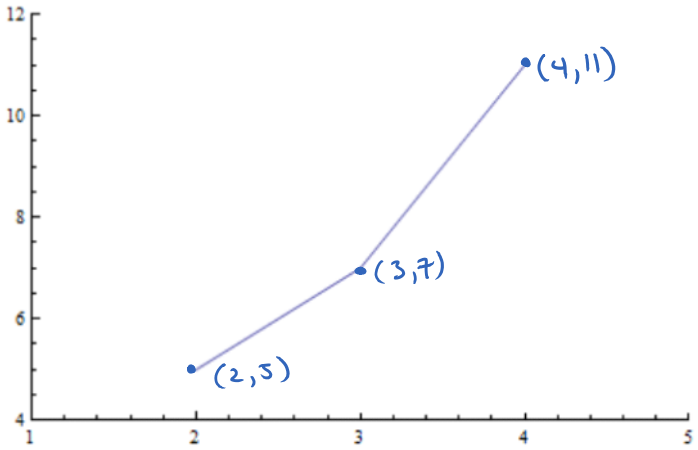
all 1s to account for the affine shift a | x-values | y-values

Example: Find the equation for the line $y = a + bx$ that goes through the points

(2,5)
(3,7)
(4,11).

Answer: There is no such line!

```
ListPlot[{{2, 5}, {3, 7}, {4, 11}},
 PlotRange → {{1, 5}, {4, 12}}, Joined → True]
```



Let's try setting it up anyway...

$$\begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \\ 11 \end{pmatrix}$$

... an infeasible set of equations

But we can minimize the squared error with

```
>> pinv([1 2; 1 3; 1 4]) * [5;7;11]

ans =

   -1.3333
    3.0000
```

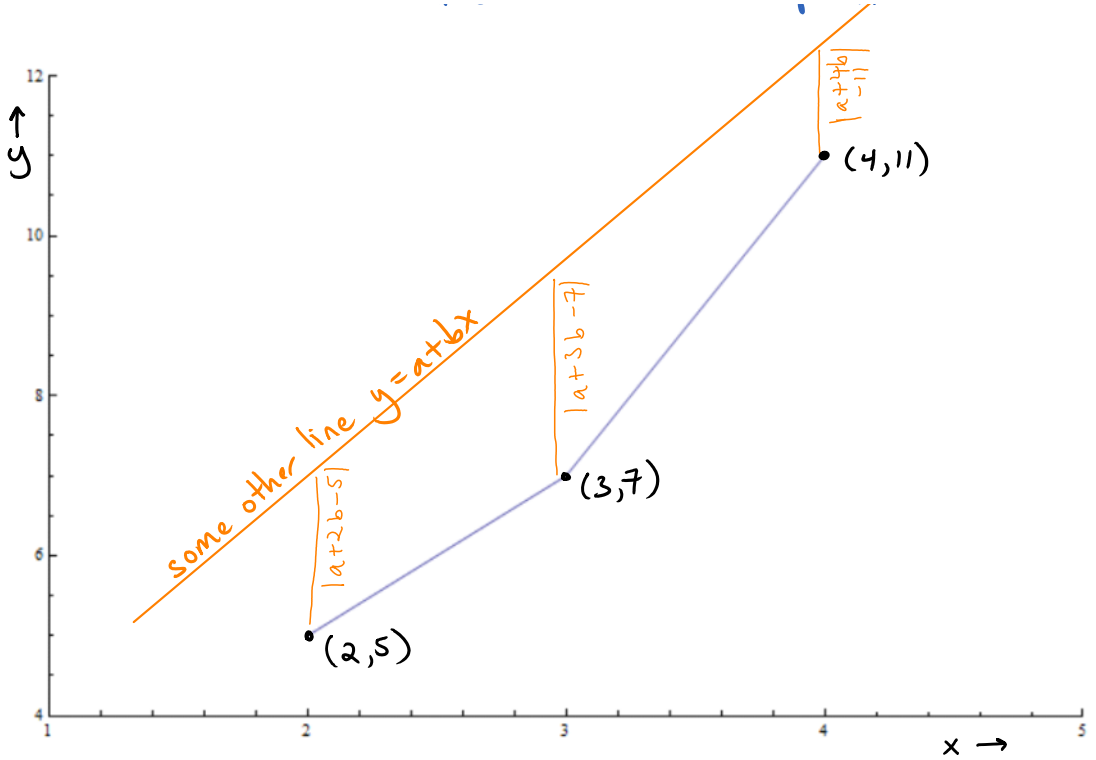
$\Rightarrow (a,b) = (-\frac{4}{3}, 3)$ minimizes $\left\| \begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} - \begin{pmatrix} 5 \\ 7 \\ 11 \end{pmatrix} \right\|^2$

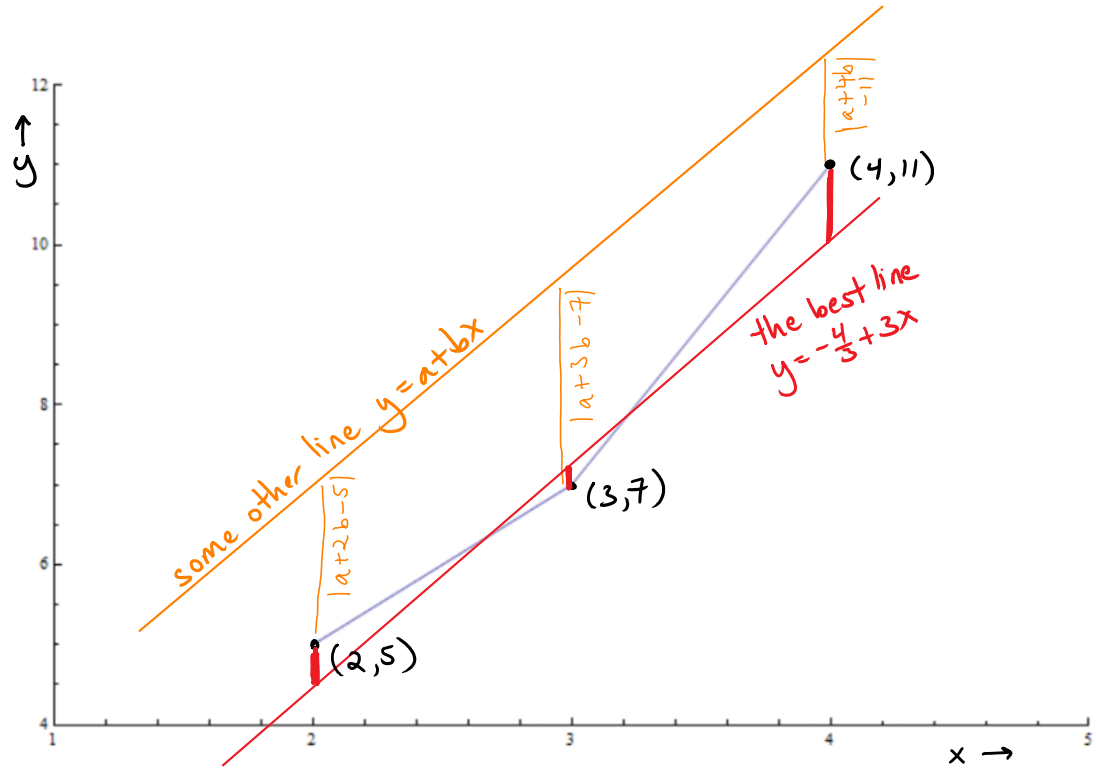
$= |a+2b-5|^2$
$+ |a+3b-7|^2$
$+ |a+4b-11|^2$

the sum of the squared errors to each data point

$|ak+b-y|$

12



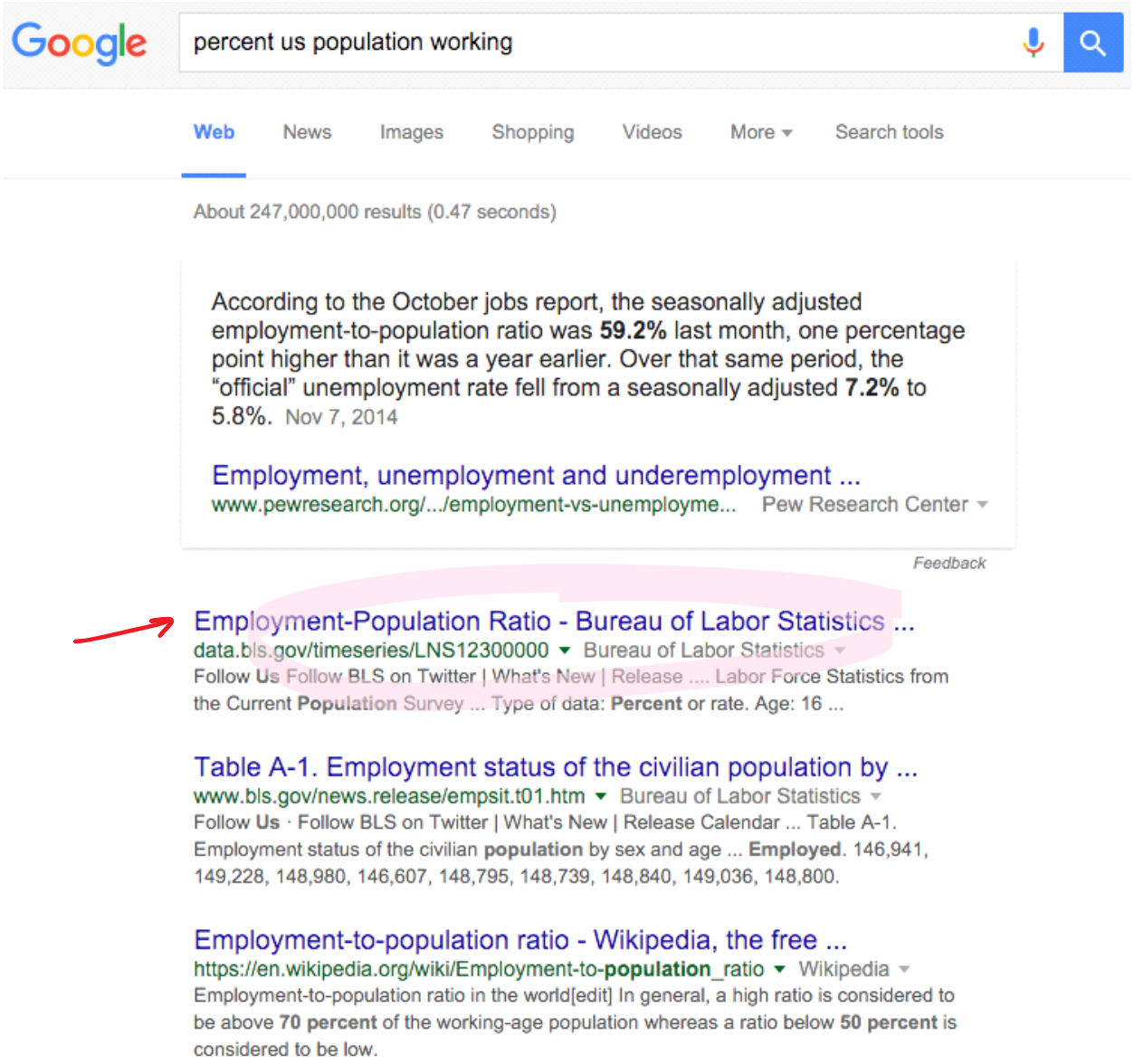


Example:

Predict what percent of Americans 16 or older will be employed in 2050.

Answer:

① Get the data.


Google
percent us population working
Web
News
Images
Shopping
Videos
More
Search tools
About 247,000,000 results (0.47 seconds)
According to the October jobs report, the seasonally adjusted employment-to-population ratio was 59.2% last month, one percentage point higher than it was a year earlier. Over that same period, the "official" unemployment rate fell from a seasonally adjusted 7.2% to 5.8%. Nov 7, 2014
Employment, unemployment and underemployment ...
www.pewresearch.org/.../employment-vs-unemployme... Pew Research Center
Feedback
Employment-Population Ratio - Bureau of Labor Statistics ...
data.bls.gov/timeseries/LNS12300000 Bureau of Labor Statistics
Follow Us Follow BLS on Twitter | What's New | Release .... Labor Force Statistics from the Current Population Survey ... Type of data: Percent or rate. Age: 16 ...
Table A-1. Employment status of the civilian population by ...
www.bls.gov/news.release/empsit.t01.htm Bureau of Labor Statistics
Follow Us · Follow BLS on Twitter | What's New | Release Calendar ... Table A-1. Employment status of the civilian population by sex and age ... Employed. 146,941, 149,228, 148,980, 146,607, 148,795, 148,739, 148,840, 149,036, 148,800.
Employment-to-population ratio - Wikipedia, the free ...
https://en.wikipedia.org/wiki/Employment-to-population_ratio Wikipedia
Employment-to-population ratio in the world[edit] In general, a high ratio is considered to be above 70 percent of the working-age population whereas a ratio below 50 percent is considered to be low.

UNITED STATES DEPARTMENT OF LABOR

BUREAU OF LABOR STATISTICS

Home | Subjects | Data Tools | Publications | Economic Releases | Students | Beta

Databases, Tables & Calculators by Subject

Change Output Options: From: 2005 To: 2015 GO

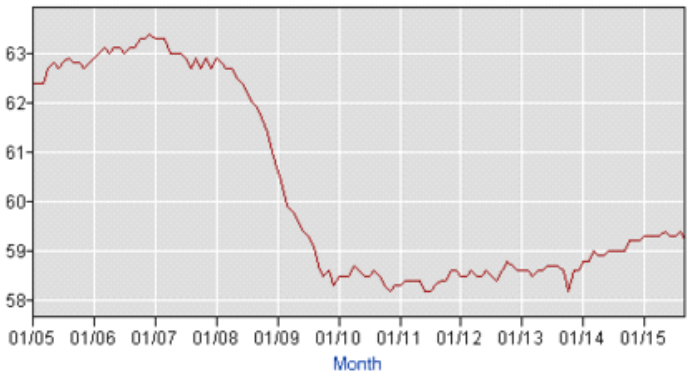
include graphs include annual averages

1948

Data extracted on: October 27, 2015 (10:44:11 AM)

Labor Force Statistics from the Current Population Survey

Series Id: LNS12300000
Seasonally Adjusted
Series title: (Seas) Employment-Population Ratio
Labor force status: Employment-population ratio
Type of data: Percent or rate
Age: 16 years and over



Download: xlsx

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2005	62.4	62.4	62.4	62.7	62.8	62.7	62.8	62.9	62.8	62.8	62.7	62.8
2006	62.9	63.0	63.1	63.0	63.1	63.1	63.0	63.1	63.1	63.3	63.3	63.4
2007	63.3	63.3	63.3	63.0	63.0	63.0	62.9	62.7	62.9	62.7	62.9	62.7
2008	62.9	62.8	62.7	62.7	62.5	62.4	62.2	62.0	61.9	61.7	61.4	61.0
2009	60.6	60.3	59.9	59.8	59.6	59.4	59.3	59.1	58.7	58.5	58.6	58.3
2010	58.5	58.5	58.5	58.7	58.6	58.5	58.5	58.6	58.5	58.3	58.2	58.3
2011	58.3	58.4	58.4	58.4	58.4	58.2	58.2	58.3	58.4	58.4	58.6	58.6
2012	58.5	58.5	58.6	58.5	58.5	58.6	58.5	58.4	58.6	58.8	58.7	58.6

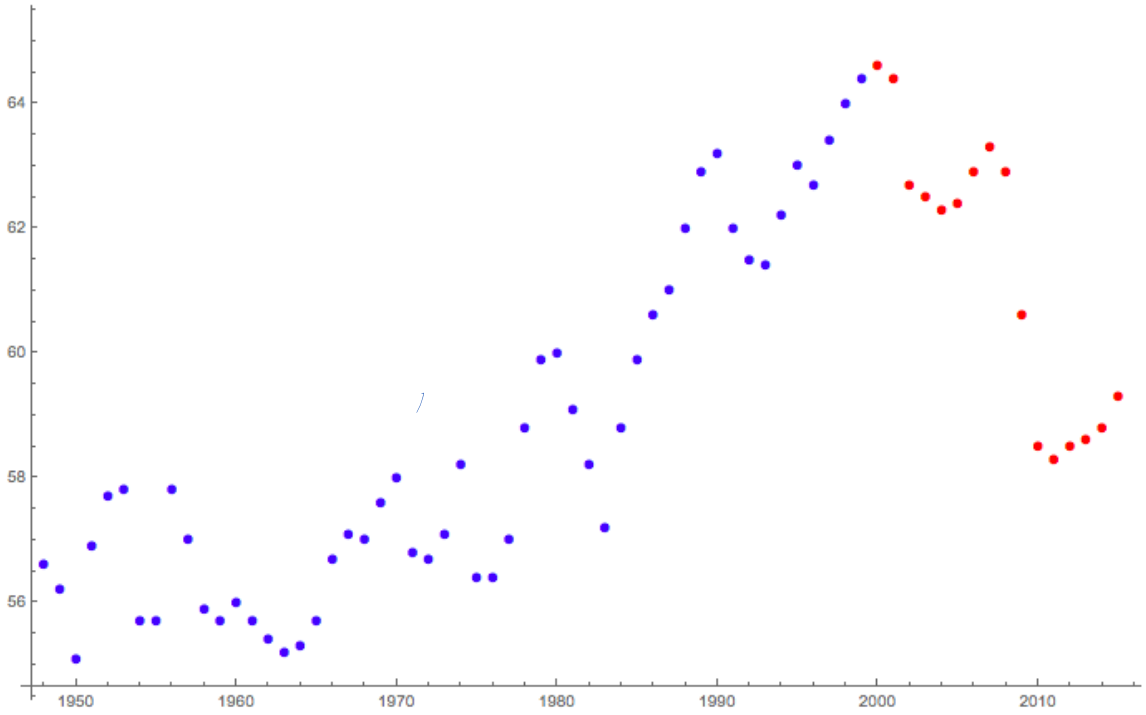
② Load the data into Mathematica / Matlab, and set up the matrices

```
data = Import["employment-population.csv"]
plot1 = ListPlot[data[[;; -17]], PlotMarkers → {Automatic, Small}, PlotStyle → Blue];
plot2 = ListPlot[data[[-16 ;;]], PlotMarkers → {Automatic, Small}, PlotStyle → Red];
plot = Show[plot1, plot2, PlotRange → {{1948, 2015}, {55, 65}}]
```

```
{{1948, 56.6}, {1949, 56.2}, {1950, 55.1}, {1951, 56.9}, {1952, 57.7}, {1953, 57.8}, {1954, 55.7}, {1955, 55.7}, {1956, 57.8}, {1957, 57.
 {1958, 55.9}, {1959, 55.7}, {1960, 56.}, {1961, 55.7}, {1962, 55.4}, {1963, 55.2}, {1964, 55.3}, {1965, 55.7}, {1966, 56.7}, {1967, 57.1
 {1968, 57.}, {1969, 57.6}, {1970, 58.}, {1971, 56.8}, {1972, 56.7}, {1973, 57.1}, {1974, 58.2}, {1975, 56.4}, {1976, 56.4}, {1977, 57.},
 {1978, 58.8}, {1979, 59.9}, {1980, 60.}, {1981, 59.1}, {1982, 58.2}, {1983, 57.2}, {1984, 58.8}, {1985, 59.9}, {1986, 60.6}, {1987, 61.}
 {1988, 62.}, {1989, 62.9}, {1990, 63.2}, {1991, 62.}, {1992, 61.5}, {1993, 61.4}, {1994, 62.2}, {1995, 63.}, {1996, 62.7}, {1997, 63.4},
 {1998, 64.}, {1999, 64.4}, {2000, 64.6}, {2001, 64.4}, {2002, 62.7}, {2003, 62.5}, {2004, 62.3}, {2005, 62.4}, {2006, 62.9},
 {2007, 63.3}, {2008, 62.9}, {2009, 60.6}, {2010, 58.5}, {2011, 58.3}, {2012, 58.5}, {2013, 58.6}, {2014, 58.8}, {2015, 59.3}}
```

```
data = Import["employment-population.csv"]
plot1 = ListPlot[data[[ ;; -17]], PlotMarkers → {Automatic, Small}, PlotStyle → Blue];
plot2 = ListPlot[data[[-16 ;;]], PlotMarkers → {Automatic, Small}, PlotStyle → Red];
plot = Show[plot1, plot2, PlotRange → {{1948, 2015}, {55, 65}}]
```

```
{{1948, 56.6}, {1949, 56.2}, {1950, 55.1}, {1951, 56.9}, {1952, 57.7}, {1953, 57.8}, {1954, 55.7}, {1955, 55.7}, {1956, 57.8}, {1957, 57.
 {1958, 55.9}, {1959, 55.7}, {1960, 56.}, {1961, 55.7}, {1962, 55.4}, {1963, 55.2}, {1964, 55.3}, {1965, 55.7}, {1966, 56.7}, {1967, 57.1
 {1968, 57.}, {1969, 57.6}, {1970, 58.}, {1971, 56.8}, {1972, 56.7}, {1973, 57.1}, {1974, 58.2}, {1975, 56.4}, {1976, 56.4}, {1977, 57.},
 {1978, 58.8}, {1979, 59.9}, {1980, 60.}, {1981, 59.1}, {1982, 58.2}, {1983, 57.2}, {1984, 58.8}, {1985, 59.9}, {1986, 60.6}, {1987, 61.}
 {1988, 62.}, {1989, 62.9}, {1990, 63.2}, {1991, 62.}, {1992, 61.5}, {1993, 61.4}, {1994, 62.2}, {1995, 63.}, {1996, 62.7}, {1997, 63.4},
 {1998, 64.}, {1999, 64.4}, {2000, 64.6}, {2001, 64.4}, {2002, 62.7}, {2003, 62.5}, {2004, 62.3}, {2005, 62.4}, {2006, 62.9},
 {2007, 63.3}, {2008, 62.9}, {2009, 60.6}, {2010, 58.5}, {2011, 58.3}, {2012, 58.5}, {2013, 58.6}, {2014, 58.8}, {2015, 59.3}}
```




```
A = data[[-16 ;;, 1]];
A = {1, #} & /@ A
b = data[[-16 ;;, 2]]
```

← insert 1's in each row

```
{{1, 2000}, {1, 2001}, {1, 2002}, {1, 2003}, {1, 2004}, {1, 2005}, {1,|2006}, {1, 2007},
 {1, 2008}, {1, 2009}, {1, 2010}, {1, 2011}, {1, 2012}, {1, 2013}, {1, 2014}, {1, 2015}}

{64.6, 64.4, 62.7, 62.5, 62.3, 62.4, 62.9, 63.3, 62.9, 60.6, 58.5, 58.3, 58.5, 58.6, 58.8, 59.3}
```

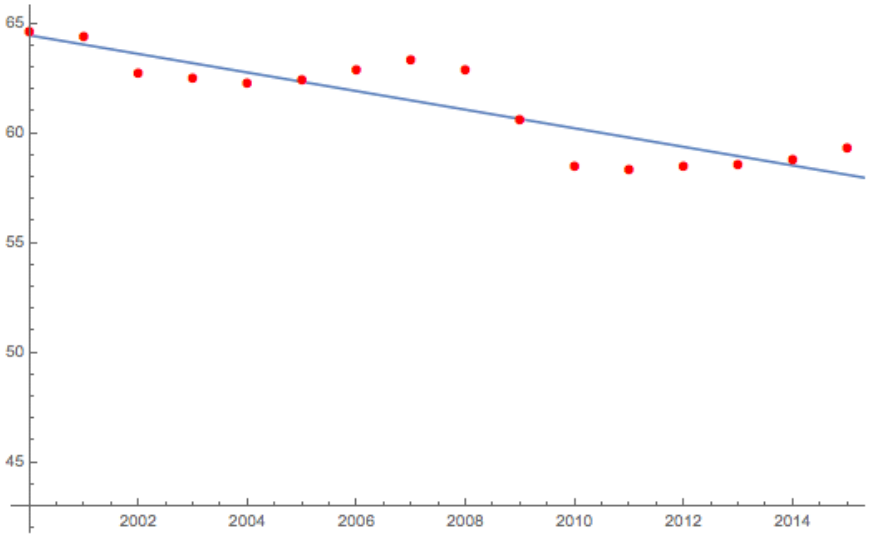
③ Compute the best-fitting line

```
{intercept, slope} = PseudoInverse[A].b
```

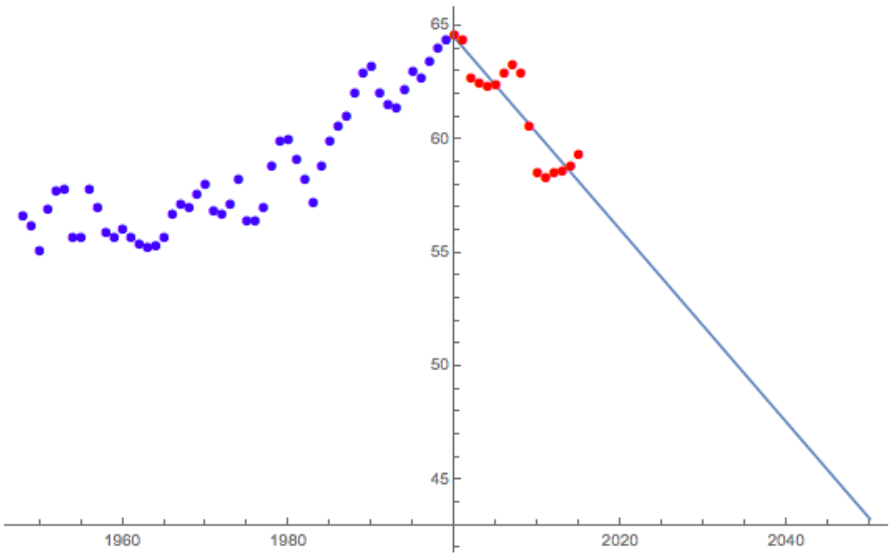
```
{912.113, -0.423824}
```

④ Evaluate it, and get the answer!

```
plot3 = Plot[intercept + slope x, {x, 2000, 2050}];
plot4 = Show[plot3, plot, PlotRange → {{2000, 2015}, Automatic}]
```




```
Show[plot4, PlotRange → {{1948, 2050}, Automatic}]
```




```
intercept + slope 2050

intercept + slope 1900

intercept + slope 2200
43.275   in 2050
106.849  in 1900
-20.2985 in 2200
```

Extrapolations often don't work well. ☺

LET'S GENERALIZE!

Setting: m data points

$$\left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \ldots, x_k^{(1)}, y^{(1)}\right)$$
$$\left(x_1^{(2)}, x_2^{(2)}, \ldots, x_k^{(2)}, y^{(2)}\right)$$
$$\vdots$$
$$\left(x_1^{(m)}, x_2^{(m)}, \ldots, x_k^{(m)}, y^{(m)}\right)$$

$x_1^{(m)}, x_2^{(m)}, \ldots, x_k^{(m)}$: components known exactly, e.g., date/time

$y^{(m)}$: component that we want to predict

Goal: Find the best linear predictor for y,

$$a_0, a_1, a_2, \ldots, a_k \in \mathbb{R}$$

to minimize total squared error.

$$\sum_{j=1}^{m} \left| a_0 + a_1 x_1^{(j)} + \cdots + a_k x_k^{(j)} - y^{(j)} \right|^2$$

Answer:

$$\underbrace{\begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_k^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_k^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & \cdots & x_k^{(3)} \\ & \vdots & & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_k^{(m)} \end{pmatrix}}_{A} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} = \underbrace{\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{pmatrix}}_{\vec{y}}$$

The least-squares solution is $A^+ \vec{y}$.

(A^+: pseudoinverse)

Example:

Consider the time (T) it takes for a runner to complete a marathon (26 miles and 385 yards). Many factors such as height, weight, age, previous training, etc. can influence an athlete's performance, but experience has shown that the following three factors are particularly important:

$$x_1 = \text{Ponderal index} = \frac{\text{height (in.)}}{[\text{weight (lbs.)}]^{\frac{1}{3}}},$$

$$x_2 = \text{Miles run the previous 8 weeks},$$

$$x_3 = \text{Age (years)}.$$

A linear model hypothesizes that the time T (in minutes) is given by $T = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \varepsilon$, where ε is a random function accounting for all other factors and whose mean value is assumed to be zero. On the basis of the five observations given below, estimate the expected marathon time for a 43-year-old runner of height 74 in., weight 180 lbs., who has run 450 miles during the previous eight weeks.

T	x_1	x_2	x_3
181	13.1	619	23
193	13.5	803	42
212	13.8	207	31
221	13.1	409	38
248	12.5	482	45

What is your personal predicted mean marathon time?

Answer: Start with a sanity check: $\alpha_1 < 0$, $\alpha_2 < 0$, $\alpha_3 > 0$

① Enter the data ② Find best-fit plane ③ Predict!

```
>> A = [
1 13.1 619 23 ;
1 13.5 803 42 ;
1 13.8 207 31 ;
1 13.1 409 38 ;
1 12.5 482 45
]; x1 x2 x3
>> b = [
181; 193; 212; 221; 248
];
```

times in minutes

```
>> alpha = pinv(A) * b

alpha =

  492.0442
  -23.4355
   -0.0761
    1.8624
```

every year older = ~2 minutes slower

2 miles more/week = 1 min. faster

```
                                  x1      x2   x3
>> alpha' * [1; 74/180^(1/3); 450; 43]

ans =

  230.7209
```

← predicted time = 3 hours 50 minutes

Example (continued): Same problem, but now fit a quadratic curve to x_2 (distance over previous 8 weeks) — running too much slows you down?

Answer: Want $T = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_2^2$

① Enter the data & find best-fit hyperplane

```
>> A2 = [A, A(:,3).^2];
alpha2 = pinv(A2) * b

alpha2 =

  740.7697
  -38.7075
   -0.2441
    1.5801
    0.0002
```

② Predict!

```
>> alpha2' * [1; 74/180^(1/3); 450; 43; 450^2]

ans =

  224.3594
```

= 3 hours 44 minutes

(Of course, be careful of overfitting the data by using too many parameters.)

(data by using too many parameters.)

Fitting other curves than lines:

Example: Predict US gross domestic product (GDP) in 2050.

Answer: Google "US GDP"

An exponential should fit the data better than a straight line

$$G = e^{a+bT}$$

$$\Rightarrow \log G = a + bT$$

years = [1966:2014];
GDP = [3800176644539.76, 3895181060653.26, 4082149751564.6, 4208696393863.11, 4343661175265.73, 4486805518455.5, 4722957883148.85, 4989480292992.24, 4963676968025.7, 4953864844037.75, 5220684465530.35, 5461284794411.99, 5765024247749.54, 5948103589527.53, 5933554752676.08, 6087499073703.77, 5971173597636.82, 6247785657762.76, 6701317851694.21, 6985369125864.67, 7230668360903.64, 7480975855992.56, 7795473984765.14, 8082388278264.36, 8237519238199.96, 8231416510730, 8524075976250.96, 8758134889170.93, 9111757146664.33, 9359503617414.74, 9714779258395.62, 10150683977472.6, 10602380376634.4, 11099123060521.4, 11553318760428.6, 11666077052746.7, 11874448085025.7, 12207737238841, 12669890778469.9, 13093726000000, 13442886679117.7, 13681977860942.7, 13642078277526.4, 13263438360402.5, 13599258090661.7, 13817044044776, 14137749306899.7, 14451509511869.6, 14796640462032];

```
>> A = [ones(length(years),1) years']

A =

     1    1966
     1    1967
     1    1968
     1    1969
     1    1970
     1    1971
     1    1972
```

```
>> b = log(GDP)'

>> x = pinv(A) * b

x =

  -29.0848
    0.0295
```

$$\begin{pmatrix} 1 & 1966 \\ 1 & 1967 \\ 1 & 1968 \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \log(5.2\times10^{11}) \\ \log(5.4\times10^{11}) \\ \log(5.8\times10^{11}) \\ \vdots \end{pmatrix}$$

```
>> x = A \ b

x =

  -29.0848
    0.0295
```

```
>> exp([1 2050] * x)

ans =

   4.7132e+13
```

= $47 trillion

Also easy to fit other functions, e.g., $a + bT + cT^2 = \log G$:

```
>> B = [A (years') .^2]
```

```
>> y = B \ b
```

```
>> exp([1 2050 2050^2] * y)
```

```
>> B = [A (years') .^2]

B =

     1        1966     3865156
     1        1967     3869089
     1        1968     3873024
     1        1969     3876961
     1        1970     3880900
     1        1971     3884841
     1        1972     3888784
```

```
>> y = B \ b

y =

  -588.2974
     0.5916
    -0.0001
```

```
>> exp([1 2050 2050^2] * y)

ans =

   2.9161e+13
```

Four ways to find x to minimize $\|Ax-b\|$:

0. Compute $P_{R(A)}\vec{b}$ (e.g., using Gram-Schmidt to get on. basis for $R(A)$)

   Solve $A\vec{x} = P_{R(A)}\vec{b}$

1. Compute pseudoinverse A^+, set $\vec{x} = A^+\vec{b}$.

   Pros: Easy to remember, one line pinv(A)*b in Matlab

   Cons: Slow for large A, numerically unstable

> Meyer p.423: **Caution!** Generalized inverses are useful in formulating theoretical statements such as those above, but, just as in the case of the ordinary inverse, generalized inverses are not practical computational tools. In addition to being computationally inefficient, serious numerical problems result from the fact that $\mathbf{A}^+$ need not be a continuous functions of the entries of A.

2. Find x manually, with calculus:

$$E = \text{squared error} = \left\| \begin{pmatrix} 1 & a_1 \\ 1 & a_2 \\ 1 & a_3 \\ \vdots & \\ 1 & a_m \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \right\|^2$$

$$= \sum_{j=1}^{m} (x_1 + a_j x_2 - b_j)^2$$

$$\frac{\partial}{\partial x_1} = 2\sum_j (x_1 + a_j x_2 - b_j) = 0 \quad \Rightarrow m x_1 + \left(\sum_j a_j\right) x_2 = \left(\sum_j b_j\right)$$

$$\frac{\partial E}{\partial x_2} = 2\sum_j a_j (x_1 + a_j x_2 - b_j) = 0 \quad \Rightarrow \left(\sum_j a_j\right) x_1 + \left(\sum_j a_j^2\right) x_2 = \sum_j a_j b_j$$

Let $\bar{a} = \frac{1}{m}\sum a_j$ (average value of a_j's)

$\bar{b} = \frac{1}{m}\sum_j b_j$ (average of b_j's

First equation

$\Rightarrow$ intercept $x_1 = \bar{b} - \bar{a}x_2$

(this is 0 if the data is centered so $\bar{a} = \bar{b} = 0$)

Second equation

$$\Rightarrow x_2 = \frac{\sum_j a_j b_j - m \cdot \bar{a} \cdot \bar{b}}{\sum_j a_j^2 - m(\bar{a})^2}$$

$$\left(= \frac{Cov(A,B)}{Var(A)} \quad \text{if } A \text{ and } B \text{ are random variables with } Pr[(A,B) = (a_j, b_j)] = \frac{1}{m}. \right)$$

Pros: None (okay, it gives an easy closed-form solution)

③ Solve $A^TAx = A^Tb$

Theorem: Let A be an $m \times n$ real matrix with rank n. Then for any b,

- The equation $A^TAx = A^Tb$ is feasible (i.e., it has a solution x)
- The unique solution is

$$(A^TA)^{-1}A^Tb = A^+b$$

(A^+: pseudoinverse)

Proof: There is a solution because

- $R(A^TA) = R(A^T)$ (shown above)
- $A^Tb \in R(A^T)$

$\Rightarrow A^Tb \in R(A^TA)$ ✓

The solution is unique because $N(A) = \{\vec{0}\}$

(because $\text{rank}(A) = \dim R(A^T) = n$, by assumption, and by rank-nullity $\dim R(A^T) + \dim N(A) = n$)

So why is $(A^TA)^{-1}A^Tb = A^+b$, as claimed? There are several ways of seeing it.

Algebraically: Using the SVD of A, $A = \sum_i \lambda_i \vec{u}_i \vec{v}_i^T$,

$$A^+ = \sum_{i:\lambda_i>0} \frac{1}{\lambda_i} v_i u_i^T$$

$$(A^TA)^{-1}A^T = \left[\sum_i \lambda_i^2 v_i v_i^T\right]^{-1}\left(\sum_j \lambda_j v_j u_j^T\right)$$

$$= \left(\sum_i \frac{1}{\lambda_i^2} v_i v_i^T\right)\left(\sum_j \lambda_j v_j u_j^T\right)$$

* since $\text{rank}(A) = n$, $\lambda_1, \ldots, \lambda_n$ are all > 0

$$= \sum_i \frac{1}{\lambda_i} v_i u_i^T$$

$$= A^+ \quad \checkmark$$

More geometrically:

$x = A^+b$ solves $Ax = P_{R(A)}b$

$\Rightarrow A^TAx = A^TP_{R(A)}b$

$= A^Tb$ since $A^T = A^TP_{R(A)}$

(in general $A = P_{R(A)}AP_{R(A^T)}$) □

Corollary: For any $m \times n$ real matrix A, of rank n,

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

Proof: Since we just showed $A^+b = (A^TA)^{-1}A^Tb$ for any b. □

Pros: Fast

Cons: condition number of A^TA = (condition number of A)^2

Method ④ Solve $\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$

Pros: Maintains sparsity of A.

[Problem 4.6.9, p.237 of Meyer]

⑤ Latest research has focused on finding fast randomized approximation algorithms, based on dimension reduction

à la Johnson-Lindenstrauss Lemma, eg.,

[Clarkson Woodruff 2012 http://arxiv.org/abs/1207.6365]

[Nelson & Nguyen 2012 http://arxiv.org/abs/1211.1002]

Lecture 16: Least squares (continued)

Admin:

"LEAST SQUARES" FITTING & APPLICATIONS TO DATA ANALYSIS

Reading:
Meyer 4.6
5.13-14
Strang 3.3

Problem: System of equations

$$A\vec{x} = \vec{b}$$

but $\vec{b} \notin R(A)$!

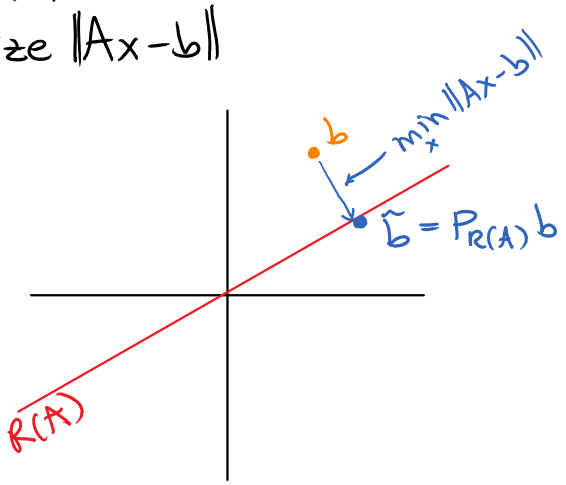
$\Rightarrow$ Find x to minimize $\|Ax-b\|$

Algebraically:

$$A = \sum_i \sigma_i \vec{u}_i \vec{v}_i^T$$

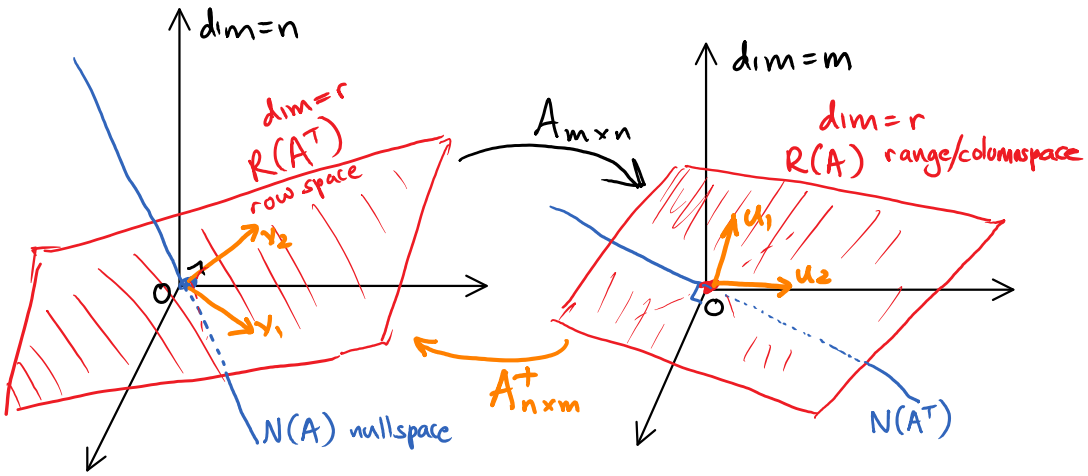
$$A^+ = \sum_{i:\sigma_i>0} \frac{1}{\sigma_i} \vec{v}_i \vec{u}_i^T$$

(psendo-inverse)



$$\Rightarrow x = A^+ b$$

minimizes $\|Ax-b\|$

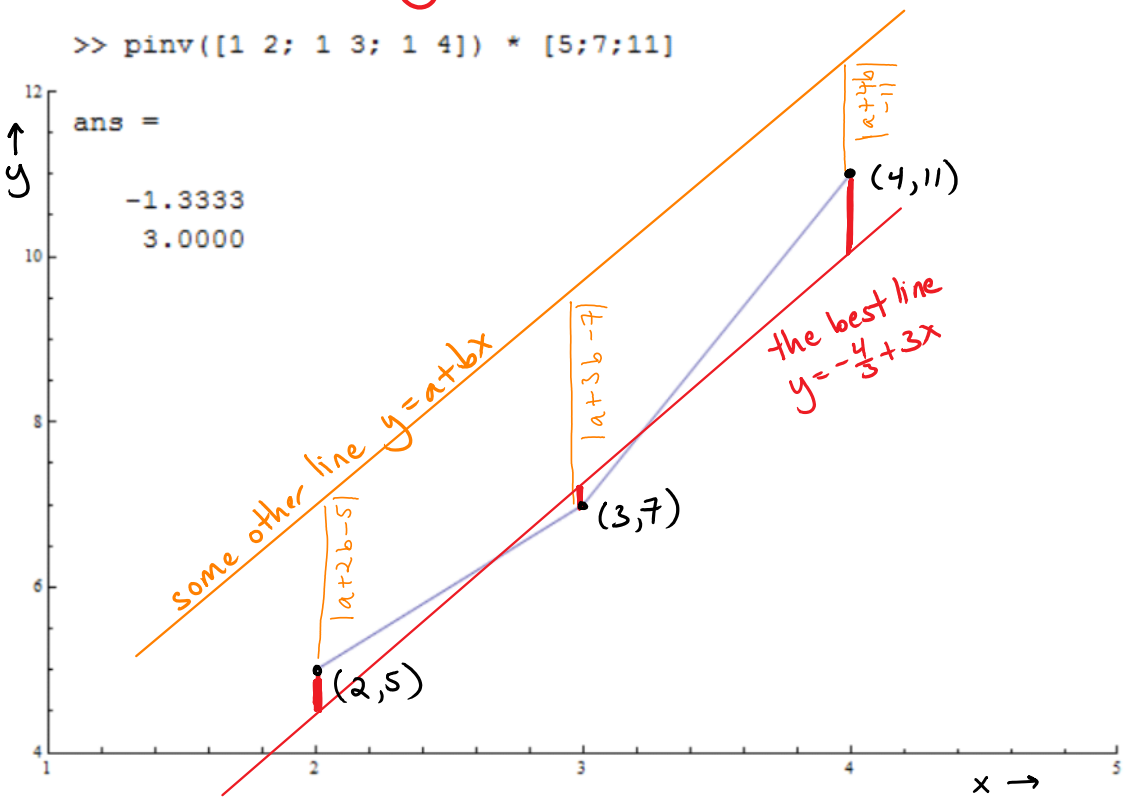


Example: Linear regression

```
>> pinv([1 2; 1 3; 1 4]) * [5;7;11]

ans =

   -1.3333
    3.0000
```



Setting: m data points

$$(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_k^{(1)}, y^{(1)})$$
$$(x_1^{(2)}, x_2^{(2)}, \dots, x_k^{(2)}, y^{(2)})$$
$$\vdots$$
$$(x_1^{(m)}, x_2^{(m)}, \dots, x_k^{(m)}, y^{(m)})$$

components known exactly, e.g., date/time

component that we want to predict

Goal: Find the best linear predictor for y,

$$a_0, a_1, a_2, \dots, a_k \in \mathbb{R}$$

to minimize total squared error.

$$\sum_{j=1}^{m} \left| a_0 + a_1 x_1^{(j)} + \cdots + a_k x_k^{(j)} - y^{(j)} \right|^2$$

Answer:

$$\begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_k^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_k^{(2)} \\ & & & & \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \end{pmatrix}$$

$$\begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_k^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_k^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & \cdots & x_k^{(3)} \\ & \vdots & & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_k^{(m)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

$$A \qquad\qquad \vec{y}$$

The least-squares solution is $A^+ \vec{y}$.

(A^+: pseudoinverse)

Evaluating the quality of a fit:

Of course there is a lot of statistics to consider when running linear regressions.

The most basic statistic is the "R^2 value", also known as the "coefficient of determination."

Definition: Given data points $(x_1, y_1), \ldots, (x_m, y_m)$, and the least-squared-error fit line $y = \alpha + \beta x$,

$$R^2 = 1 - \frac{\sum_j (y_j - \alpha - \beta x_j)^2}{\sum_j (y_j - \bar{y})^2}$$

where $\bar{y} = \frac{1}{m}\sum_j y_j$

$$= 1 - \frac{(\text{total squared error from best-fit line})}{(\text{total squared error from horizontal line } \bar{y})}$$

Observe: $0 \le R^2 \le 1$

and higher is better

- Collinear data points will have $R^2 = 1$ (since best-fit line will fit exactly)
- But what counts as a "good fit" depends very much on the field (eg., macroeconomics)

Interpretation of R^2: Explained variance

If the data were distributed randomly about the mean $\bar{y}$,

the observed variance is $\mathbb{E}[Y-\mathbb{E}(Y)]^2 = \frac{1}{m}\sum_{j=1}^{m}(y_j-\bar{y})^2$.

For data distributed randomly off the line $y=\alpha+\beta x$,

ie., $y_j = \alpha+\beta x_j + e_j$, the remainining variance is

$$\frac{1}{m}\sum_j e_j^2 = \frac{1}{m}\sum_j (y_j-\alpha-\beta x_j)^2.$$

$\Rightarrow R^2$ measures "how much of the observed variance is explained by the observed x_j data points.

Probabilistic interpretation: Why least squares?

Choosing $\vec{x}$ to minimize $\|A\vec{x}-\vec{y}\|$ is geometrically natural, but why is this natural in data analysis?

One reason (of several): In a natural probabilistic model, least-squares regression finds the maximum likelihood estimate of $\vec{x}$.

Model: Assume that the y-values are generated via

$$y^{(i)} = \vec{\alpha}\cdot\vec{x}^{(i)} + \varepsilon^{(i)}$$

where $\varepsilon^{(i)} \sim N(0,\sigma^2)$ — independent, identically distributed, Gaussian random noise.

$$\Rightarrow p_\alpha(y^{(i)}\,|\,x^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[\frac{-1}{2\sigma^2}(y^{(i)}-\alpha\cdot x^{(i)})^2\right]$$

↑ probability density function of $y^{(i)}$ for model parameters $\vec{\alpha}$

$$\Rightarrow p_\alpha(y^{(1)},\dots,y^{(m)}\,|\,x^{(1)}\dots x^{(m)}) = \frac{1}{(2\pi\sigma^2)^{m/2}}\exp\left[\frac{-1}{2\sigma^2}\sum_{i=1}^{m}(y^{(i)}-\alpha\cdot x^{(i)})^2\right]$$

$\Rightarrow$ Setting $\vec{\alpha}$ to minimize $\sum_i (y^{(i)}-\alpha\cdot x^{(i)})^2$ maximizes the likelihood of the observed data.

(**Notice**: This holds for any value of the variance σ^2, and σ need not be known to choose α.)

Linear regression with weights:

If errors between observations are correlated, or if some observations are more reliable than others, then minimizing the weighted sum of squared errors is better than unweighted.

Example: Say

$$y_1 = \alpha x_1 + \varepsilon_1 \quad , \quad \varepsilon_1 \sim N(\overset{\mu}{0}, \overset{\sigma^2}{1}) \quad \leftarrow \text{more reliable}$$

$$y_2 = \alpha x_2 + \varepsilon_2 \quad , \quad \varepsilon_2 \sim N(0, 2) \quad \leftarrow \text{less reliable}$$

$$\Rightarrow P_\alpha[y_1, y_2 | x_1, x_2] = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y_1 - \alpha x_1)^2} \cdot \frac{1}{\sqrt{2\pi \cdot 2}} e^{\frac{-1}{2\cdot 2}(y_2 - \alpha x_2)^2}$$

$$\propto \exp\left[-\frac{1}{2} \cdot \left\{(y_1 - \alpha x_1)^2 + \frac{1}{2}(y_2 - \alpha x_2)^2\right\}\right]$$

$\Rightarrow$ The max-likelihood α minimizes the weighted sum

$$S = \frac{1}{\sigma_1^2}(y_1 - \alpha x_1)^2 + \frac{1}{\sigma_2^2}(y_2 - \alpha x_2)^2$$

$$-\frac{1}{2}\frac{\partial S}{\partial \alpha} = \frac{x_1}{\sigma_1^2}(y_1 - \alpha x_1) + \frac{x_2}{\sigma_2^2}(y_2 - \alpha x_2) = 0$$

$$\Rightarrow \alpha = \frac{\frac{x_1 y_1}{\sigma_1^2} + \frac{x_2 y_2}{\sigma_2^2}}{\frac{x_1^2}{\sigma_1^2} + \frac{x_2^2}{\sigma_2^2}}$$

More generally: To choose $\vec{\alpha}$ to minimize

$$\sum_{j=1}^{m} w_j (y_j - x_j \cdot \alpha)^2,$$

observe $\| W(y - A\alpha) \|^2$ where $W = \begin{pmatrix} w_1 & & 0 \\ & w_2 & \\ & & \ddots \\ 0 & & w_m \end{pmatrix}$

$$A = \begin{pmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_m & — \end{pmatrix}$$

This is standard least-squares for the equation

$$WA\alpha = Wy$$

$\Rightarrow$ best α satisfies

$$A^T W^T W A \alpha = A^T W^T W y \text{, etc.}$$

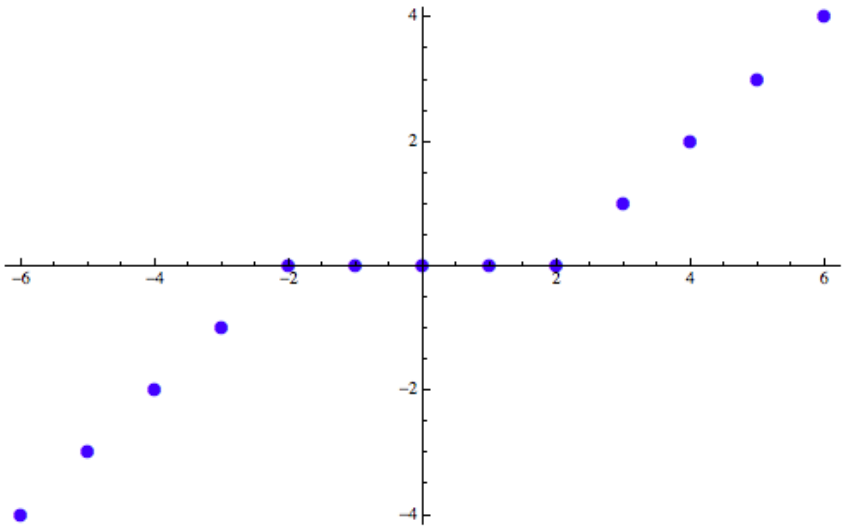
Practical application: Locally weighted linear regression

Example:

```
data = {
   {-6, -4},
```

```
data = {
    {-6, -4},
    {-5, -3},
    {-4, -2},
    {-3, -1},
    {-2, 0},
    {-1, 0},
    {0, 0},
    {1, 0},
    {2, 0},
    {3, 1},
    {4, 2},
    {5, 3},
    {6, 4}
   };
m = Length[data];

dataplot = ListPlot[data]
```



```
X = Table[{1, data[[j, 1]]}, {j, 1, m}];
X
y = data[[All, 2]];
ymean = 1/m Plus @@ y;

linear = PseudoInverse[X].y
plotlinear = Plot[{1, x}.linear, {x, -6, 6}
Show[dataplot, plotlinear]
```

```
{{1, -6}, {1, -5}, {1, -4}, {1, -3}, {1, -
 {1, -1}, {1, 0}, {1, 1}, {1, 2}, {1, 3}, {
```

$$\left\{0, \frac{50}{91}\right\}$$

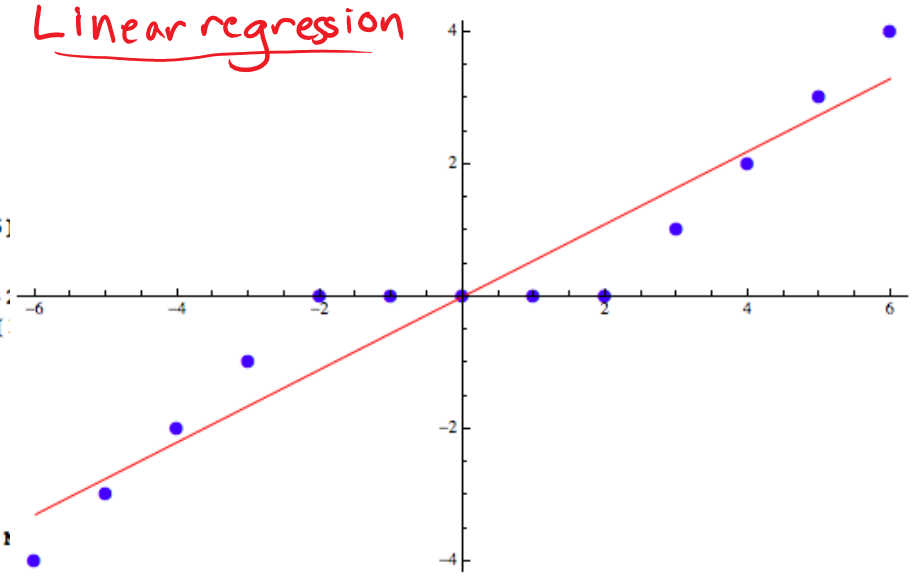
```
"R^2 value:"
```

$$R2linear = 1 - \frac{\sum_{j=1}^{m} (y[[j]] - X[[j]].linear)^2}{\sum_{j=1}^{m} (y[[j]] - ymean)^2} \text{ // } 1$$

```
R^2 value:

0.915751
```

Linear regression



Cubic regression $1, x, x^2, x^3$

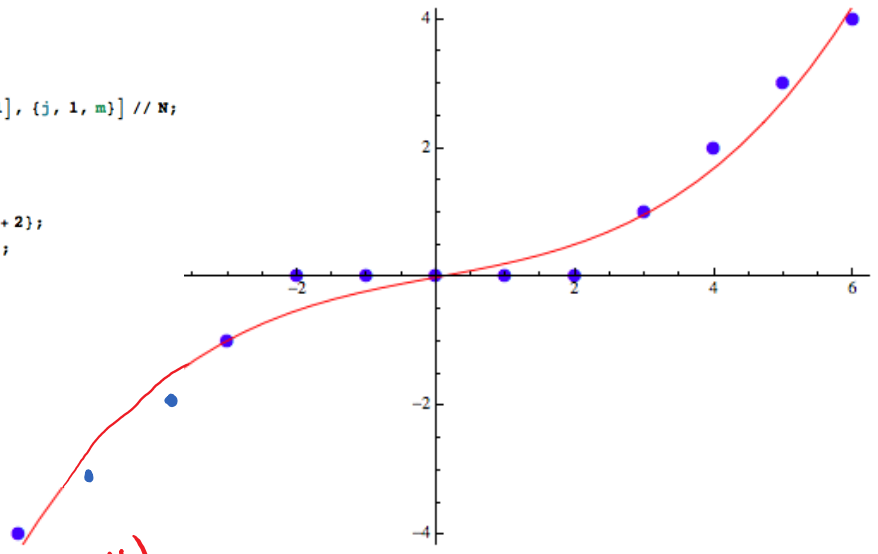
```
HigherOrderRegression[data, 3]
α= {0, 0.1998, 0, 0.013986}
R2= 0.982884
```

```
HigherOrderRegression[data_, order_] :=
  Module[{m, y, ymean, X, α, plot, min, max, R2},
   m = Length[data];
   y = data[[All, 2]];
   X = Table[Prepend[Table[data[[j, 1]]^k, {k, 1, order}], 1], {j, 1, m}] // N;

   α = PseudoInverse[X].y;
   Print["α= ", α // Chop];

   {min, max} = {Min[data[[All, 1]] - 2, Max[data[[All, 1]] + 2};
   plot = Plot[Table[x^k, {k, 0, order}].α, {x, min, max}];

   ymean = 1/m Plus @@ y;
   R2 = 1 - (Σ_{j=1}^{m} (y[[j]] - X[[j]].α)^2) / (Σ_{j=1}^{m} (y[[j]] - ymean)^2) // N;
   Print["R2= ", R2];
   Show[dataplot, plot]
  ];
```

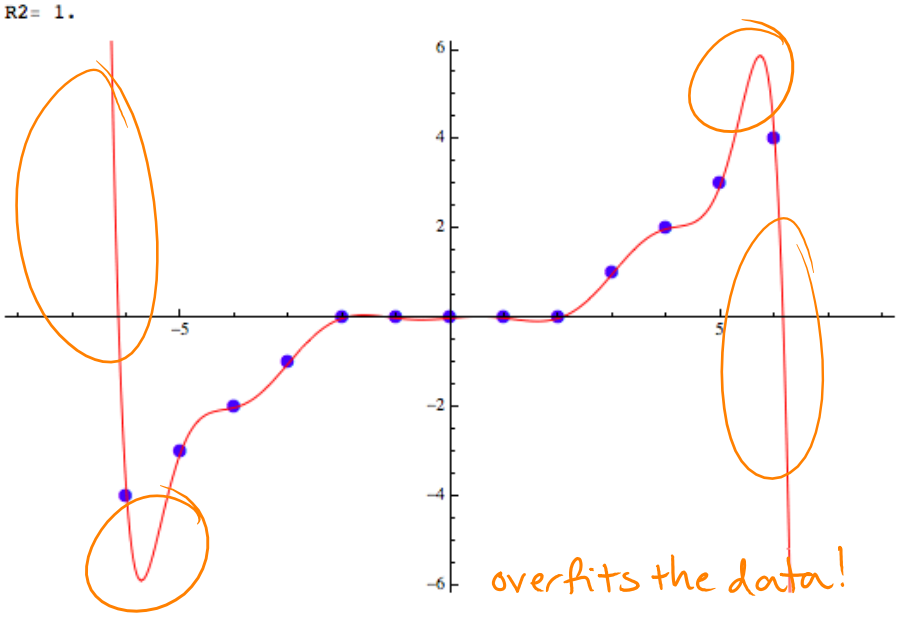


11th order $(1, x, x^2, \ldots, x^{11})$

11th order $(1, x, x^2, \ldots, x^{11})$

```
Show[HigherOrderRegression[data, 11], PlotRange → {{-8, 8}, {-6, 6}}]
```

```
α= {1.70782×10^-10, 0.101443, -2.42844×10^-10,
  -0.137535, 0, 0.03917, 0, -0.00318122, 0, 0.000104718, 0, -1.2025×10^-6}
R2= 1.
```



overfits the data!

Locally weighted linear regression

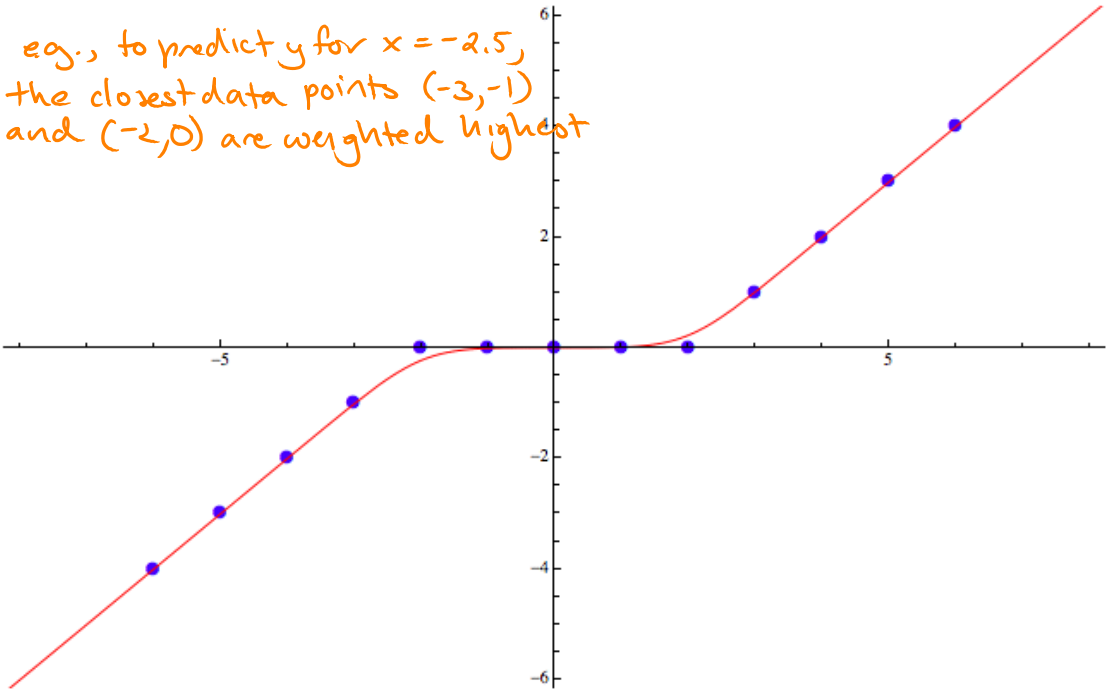
— to predict y at point x,
run weighted linear regression
giving higher weights to data points close to x !

e.g., $w_j = e^{-\frac{1}{2\tau^2}(x_j - x)^2}$

```
m = Length[data];
y = data[[All, 2]];
X = Table[{1, data[[j, 1]]}, {j, 1, m}] // N[#, 32] &;

τ = 1;

weightedlinearplot = Plot[
   W = DiagonalMatrix[Table[Exp[-1/(2 τ^2) (data[[j, 1]] - x)^2], {j, 1, m}]];
   α = PseudoInverse[W.X].W.y;
   {1, x}.α,
   {x, -10, 10}];
Show[dataplot, weightedlinearplot, PlotRange → {{-8, 8}, {-6, 6}}]
```



Note: • After running weighted linear regression at x, we don't keep the line — just the prediction at x.

- This is slower than running linear regression just once.
- Full "training set" must be kept to make predictions.
  (known as a "non-parametric" learning algorithm)