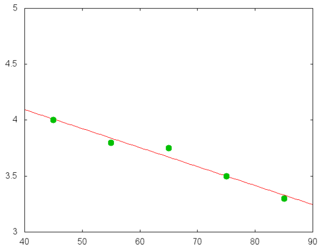
Application of least squares: linear regression

Finding the line that best fits some two-dimensional data.

Data on age versus brain mass from the Bureau of Made-up Numbers:

age	brain mass
45	4 lbs.
55	3.8
65	3.75
75	3.5
85	3.3



Let $f(x)$ be the function that predicts brain mass for someone of age x.

Hypothesis: after age 45, brain mass decreases linearly with age, i.e. that $f(x) = mx + b$ for some numbers m, b.

Goal: find m, b to as to minimize the sum of squares of prediction errors

The observations are $(x_1, y_1) = (45, 4)$, $(x_2, y_2) = (55, 3.8)$, $(x_3, y_3) = (64, 3.75)$,$(x_4, y_4) = (75, 3.5)$, $(x_5, y_5) = (85, 3.3)$.

The prediction error on the the i^{th} observation is $|f(x_i) - y_i|$.

The sum of squares of prediction errors is $\sum_i (f(x_i) - y_i)^2$.

For each observation, measure the difference between the predicted and observed y-value.

In this application, this difference is measured in pounds.

Measuring the distance from the point to the line wouldn't make sense.

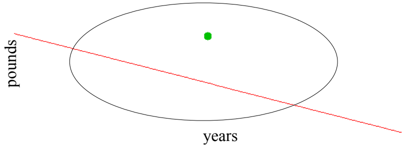
Application of least squares: linear regression

Finding the line that best fits some two-dimensional data.

Data on age versus brain mass from the Bureau of Made-up Numbers:

age	brain mass
45	4 lbs.
55	3.8
65	3.75
75	3.5
85	3.3

Let $f(x)$ be the function that predicts brain mass for someone of age x.
Hypothesis: after age 45, brain mass decreases linearly with age, i.e. that $f(x) = mx + b$ for some numbers m, b.
Goal: find m, b to as to minimize the sum of squares of prediction errors
The observations are $(x_1, y_1) = (45, 4)$, $(x_2, y_2) = (55, 3.8)$, $(x_3, y_3) = (64, 3.75)$,$(x_4, y_4) = (75, 3.5)$, $(x_5, y_5) = (85, 3.3)$.
The prediction error on the the i^{th} observation is $|f(x_i) - y_i|$.
The sum of squares of prediction errors is $\sum_i (f(x_i) - y_i)^2$.



For each observation, measure the difference between the predicted and observed y-value.
In this application, this difference is measured in pounds.
Measuring the distance from the point to the line wouldn't make sense.

Linear regression

To find the best line for given data $(x_1, y_1),(x_2, y_2),(x_3, y_3),(x_4, y_4),(x_5, y_5)$, solve this least-squares problem

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

The dot-product of row i with the vector $[m, b]$ is $mx_i + b$, i.e. the value predicted by $f(x) = mx + b$ for the i^{th} observation.

Therefore, the vector of predictions is $A \begin{bmatrix} m \\ b \end{bmatrix}$.

The vector of differences between predictions and observed values is $A \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$,

and the sum of squares of differences is the squared norm of this vector. Therefore the method of least squares can be used to find the pair (m, b) that minimizes the sum of squares, i.e. the line that best fits the data.

Application of least squares: coping with approximate data

Recall the *industrial espionage* problem: finding the number of each product being produced from the amount of each resource being consumed.

Let M =

	metal	concrete	plastic	water	electricity
garden gnome	0	1.3	.2	.8	.4
hula hoop	0	0	1.5	.4	.3
slinky	.25	0	0	.2	.7
silly putty	0	0	.3	.7	.5
salad shooter	.15	0	.5	.4	.8

We solved $\mathbf{u}^T M = \mathbf{b}$ where $\mathbf{b}$ is vector giving amount of each resource consumed:

$\mathbf{b} =$

metal	concrete	plastic	water	electricity
226.25	1300	677	1485	1409.5

`solve(M.transpose(), b)` gives us $\mathbf{u} \approx$

gnome	hoop	slinky	putty	shooter
1000	175	860	590	75

Application of least squares: industrial espionage problem

More realistic scenario: measurement of resources consumed is approximate

True amounts: $\mathbf{b} =$

metal	concrete	plastic	water	electricity
226.25	1300	677	1485	1409.5

Solving with true amounts gives

gnome	hoop	slinky	putty	shooter
1000	175	860	590	75

Measurements: $\tilde{\mathbf{b}} =$

metal	concrete	plastic	water	electricity
223.23	1331.62	679.32	1488.69	1492.64

Solving with measurements gives

gnome	hoop	slinky	putty	shooter
1024.32	28.85	536.32	446.7	594.34

Slight changes in input data leads to pretty big changes in output.

Output data not accurate, perhaps not useful! (see slinky, shooter)

Question: How can we improve accuracy of output without more accurate measurements?

Answer: More measurements!

Application of least squares: industrial espionage problem

Have to measure something else, e.g. amount of waste water produced

	metal	concrete	plastic	water	electricity	waste water
garden gnome	0	1.3	.2	.8	.4	.3
hula hoop	0	0	1.5	.4	.3	.35
slinky	.25	0	0	.2	.7	0
silly putty	0	0	.3	.7	.5	.2
salad shooter	.15	0	.5	.4	.8	.15

Measured: $\tilde{\mathbf{b}} =$

metal	concrete	plastic	water	electricity	waste water
223.23	1331.62	679.32	1488.69	1492.64	489.19

Equation $\mathbf{u} * M = \tilde{\mathbf{b}}$ is more constrained $\Rightarrow$ has no solution

but least-squares solution is

gnome	hoop	slinky	putty	shooter
1022.26	191.8	1005.58	549.63	41.1

True amounts:

gnome	hoop	slinky	putty	shooter
1000	175	860	590	75

Better output accuracy with same input accuracy

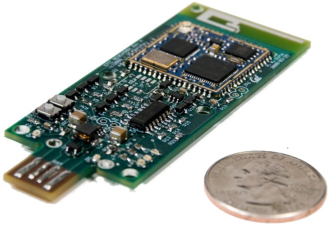
Application of least squares: Sensor node problem

Recall *sensor node problem*: estimate current draw for each hardware component
Define `D = {'radio', 'sensor', 'memory', 'CPU'}`.
Goal: Compute a `D`-vector $\mathbf{u}$ that, for each hardware component, gives the current drawn by that component.
Four test periods:

- total mA-seconds in these test periods $\mathbf{b} = [140, 170, 60, 170]$
- for each test period, vector specifying how long each hardware device was operating:
  duration$_1$ = `Vec(D, 'radio':0.1, 'CPU':0.3)`
  duration$_2$ = `Vec(D, 'sensor':0.2, 'CPU':0.4)`
  duration$_3$ = `Vec(D, 'memory':0.3, 'CPU':0.1)`
  duration$_4$ = `Vec(D, 'memory':0.5, 'CPU':0.4)`

To get $\mathbf{u}$, solve $A\mathbf{x} = \mathbf{b}$ where

$$A = \begin{bmatrix} \textbf{duration}_1 \\ \hline \textbf{duration}_2 \\ \hline \textbf{duration}_3 \\ \hline \textbf{duration}_4 \end{bmatrix}$$



Application of least squares: Sensor node problem

If measurement are exact, get back true current draw for each hardware component:

$$\mathbf{b} = [140, 170, 60, 170]$$

solve $A\mathbf{x} = \mathbf{b}$

radio	sensor	CPU	memory
500	250	300	100

More realistic: approximate measurement

$$\tilde{\mathbf{b}} = [141.27, 160.59, 62.47, 181.25]$$

solve $A\mathbf{x} = \tilde{\mathbf{b}}$

radio	sensor	CPU	memory
421	142	331	98.1

How can we get more accurate results?

Solution: Add more test periods and solve least-squares problem

Application of least squares: Sensor node problem

$\textbf{duration}_1$ = `Vec(D, 'radio':0.1, 'CPU':0.3)`
$\textbf{duration}_2$ = `Vec(D, 'sensor':0.2, 'CPU':0.4)`
$\textbf{duration}_3$ = `Vec(D, 'memory':0.3, 'CPU':0.1)`
$\textbf{duration}_4$ = `Vec(D, 'memory':0.5, 'CPU':0.4)`
$\textbf{duration}_5$ = `Vec(D, 'radio':0.2, 'CPU':0.5)`
$\textbf{duration}_6$ = `Vec(D, 'sensor':0.3, 'radio':0.8, 'CPU':0.9, 'memory':0.8)`
$\textbf{duration}_7$ = `Vec(D, 'sensor':0.5, 'radio':0.3 'CPU':0.9, 'memory':0.5)`
$\textbf{duration}_8$ = `Vec(D, 'radio':0.2 'CPU':0.6)`

Let $A = \begin{bmatrix} \textbf{duration}_1 \\ \hline \textbf{duration}_2 \\ \hline \textbf{duration}_3 \\ \hline \textbf{duration}_4 \\ \hline \textbf{duration}_5 \\ \hline \textbf{duration}_6 \\ \hline \textbf{duration}_7 \\ \hline \textbf{duration}_8 \end{bmatrix}$

Measurement vector is $\tilde{\mathbf{b}} =$
$[141.27, 160.59, 62.47, 181.25, 247.74, 804.58, 609.10, 282.09]$

Now $A\mathbf{x} = \tilde{\mathbf{b}}$ has no solution

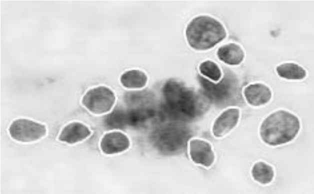
But solution to least-squares problem is

radio	sensor	CPU	memory
451.40	252.07	314.37	111.66

True solution is

radio	sensor	CPU	memory
500	250	300	100

Applications of least squares: breast cancer machine-learning problem



Recall: breast-cancer machine-learning lab

Input: vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m$ giving features of specimen, values $b_1, \ldots, b_m$ specifying $+1$ (malignant) or -1 (benign)

Informal goal: Find vector $\mathbf{w}$ such that sign of $\mathbf{a}_i \cdot \mathbf{w}$ predicts sign of b_i

Formal goal: Find vector $\mathbf{w}$ to minimize sum of squared errors
$(b_1 - \mathbf{a}_1 \cdot \mathbf{w})^2 + \cdots + (b_m - \mathbf{a}_m \cdot \mathbf{w})^2$

Approach: Gradient descent

Results: Took a few minutes to get a solution with error rate around 7%

Can we do better with least squares?

Applications of least squares: breast cancer machine-learning problem

Goal: Find the vector $\mathbf{w}$ that minimizes $(\mathbf{b}[1] - \mathbf{a}_1 \cdot \mathbf{w})^2 + \cdots + (\mathbf{b}[m] - \mathbf{a}_m \cdot \mathbf{w})^2$

Equivalent: Find the vector $\mathbf{w}$ that minimizes $\left\| \begin{bmatrix} \mathbf{b} \end{bmatrix} - \begin{bmatrix} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{bmatrix} \begin{bmatrix} \mathbf{x} \end{bmatrix} \right\|^2$

This is the least-squares problem.

Using the algorithm based on QR factorization takes a fraction of a second and gets a solution with half the error rate.

Even better solutions using more sophisticated techniques in linear algebra:

- Use an inner product that better reflects the variance of each of the features.
- Use *linear programming*
- Even more general: use *convex programming*

These are topics beyond the scope of this course. Now go learn them!