Curve-Fitting

CS 221 Lecture 13

Tue 29 November 2011

Agenda

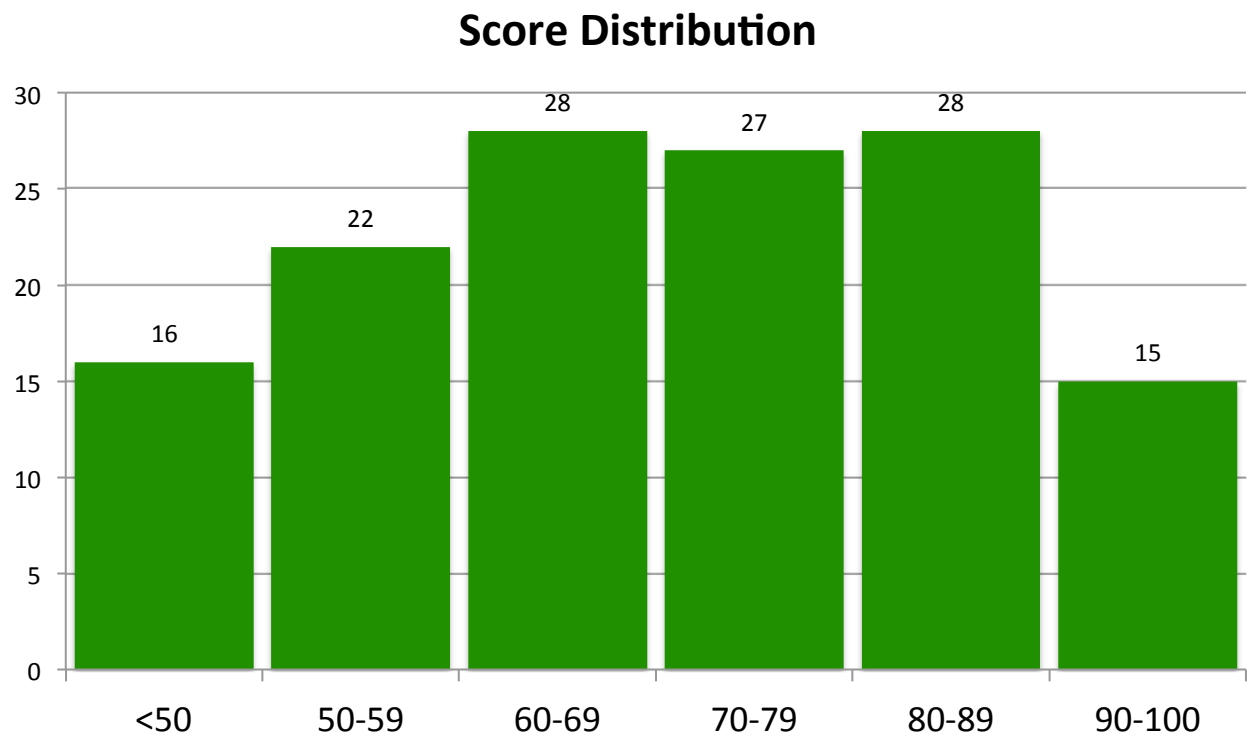
1. Announcements
2. Quiz Results
3. Least-squares regression (Curve-fitting)

   MATLAB polyval() and polyfit()
4. Examples
5. Application: Quantifying goodness-of-fit
6. Applying Linear Regression to nonlinear relationships
7. A Quick word on file I/O in MATLAB

1. Announcements

- Lab Quiz this Thursday. Coverage:
  - Solving systems of linear equations
  - Plotting data
- The End Game:
  - Prob Set 5 is out; due Sunday (but no late penalty)
  - Extra credit Prob Set on the way; due Thursday of Dead week.
  - Next week: last lecture.
    - A few odds and ends; review of what we've learned
    - Need a volunteer to administer the Teacher-Course Evaluation Survey.
- Final Exam Thursday 15 December, 10:30-12:30 HERE.

2. Class Quiz 3 Statistics



Mean = 69.1, Median = 70, Min = 23, Max = 98

N = 136

3. LEAST-SQUARES REGRESSION

The Problem

- You are given a bunch of data points (x-y pairs)
  - Measured or observed (think: experiment)
- Your job is to find (or characterize) the general relationship between x and y
  - More generally, between x_1, x_2, ..., x_n and y
- Why do this?
  - To understand the relationship
  - To interpolate values between measured values
  - To extrapolate from measured values

The Problem

Examples:

- Expansion vs. temperature of a material
- Population growth models: rate of growth of population is proportial to population size – up to a point!
- Tensile strength of a plastic material is proportional to the time it is heated
- Metabolism rate as a function of body mass
- Flow rate through a pipe as a function of diameter and slope

Things to Remember

- Given a set of measurements, there are many ways to characterize their relationship.
  - Mean, variance, other statistics
  - "Trend line" (more anon)
  - A polynomial curve that passes through the points
- Which one is best depends on
  - What you know, and
  - What you are trying to do

Example

- 8 measurements of the deflection of a wooden rod under different weights
  - weights: 1, 2, ..., 8 pounds
  - deflection in inches
- Using MATLAB's polyfit() function, we can find polynomials (including straight lines) that best "fit" a given set of (x,y) pairs.
  - p = polyfit(x,y,n): returns a degree-n polynomial that "best fits" the given set of pairs
    - p is a vector of size n+1, containing the coefficients

Defining "Best Fit"

- Suppose we want to fit a line to our data.
- Assume measured values (pairs) are all equal to function + some error:

$$y = a_0 + a_1 x + e$$

  - Note: can generalize this to degree-n polynomial:

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n + e$$

- What do we mean by "best fit"?

Definitions

- Residual: given a model (linear or polynomial) for the data, the difference between the value predicted by the model for a given x, and the measured value:

$$\text{model: } y = a_0 + a_1 x + e$$

$$e_i = y_i - \boxed{a_0 - a_1 x_i}$$

← Predicted value for given x_i

Inadequate Definition 1

- Best fit = the one that minimize the sum of the residuals:

$$\Sigma\, e_i = \Sigma\, (y_i - a_0 - a_1\, x_i)$$

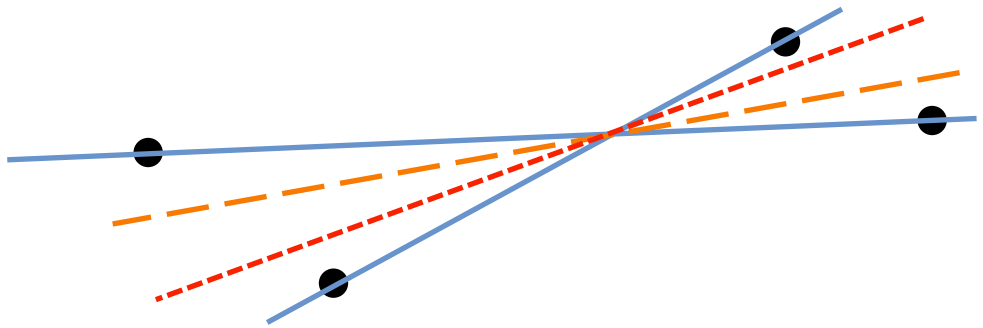
- Problem: not unique (+/- errors cancel)
  - Given two points, the best fit should be the line through those two points
  - This quantity is minimized by *any* line passing through the point halfway between the two points (except a vertical line)

Inadequate Definition 2

- Minimize the sum of the absolute value of the residuals:

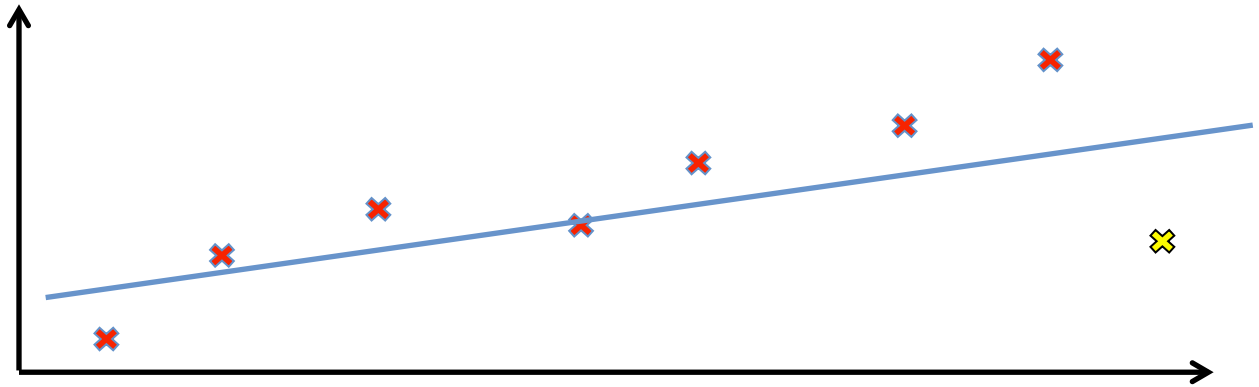
$$\Sigma\,|e_i| \;=\; \Sigma\,|y_i - a_0 - a_1\,x_i|$$

- Problem: not unique
  - Given 4 points, any line between the two lines connecting different pairs will minimize this



Inadequate Definition 3

- Minimax: minimize the maximum distance between any point and the line
- Problem: outliers
  - This strategy gives each point equal weight
  - Outliers have too much influence on the line



Least-Squares Regression

- Idea:

Minimize the sum of the squares of the residuals:

$$S_r = \Sigma e_i^2 = \Sigma (y_i - a_0 - a_1 x_i)^2$$

- This gives a unique solution.
- It can also be generalized to any polynomial:

$$S_r = \Sigma e_i^2 = \Sigma (y_i - a_0 - a_1 x_i - a_2 x_i^2 - \ldots a_n x_i^n)^2$$

Solving for Least-Squares Parameters

- For a line: two unknowns: a_0 and a_1
- Take partial derivatives w.r.t. a_0 and a_1

$$\delta S_r/\delta a_0 = -2 \Sigma (y_i - a_0 - a_1 x_i)$$

$$\delta S_r/\delta a_1 = -2 \Sigma [(y_i - a_0 - a_1 x_i) x_i]$$

- Set RHS's to zero, rearrange, and get the normal equations:

$$n a_0 + (\Sigma x_i) a_1 = \Sigma y_i$$

$$(\Sigma x_i) a_0 + (\Sigma x_i^2) a_1 = \Sigma x_i y_i$$

Formula for degree-1 (linear) Least-Squares Coefficients

- $a_1 = (n \Sigma x_i y_i - \Sigma x_i \Sigma y_i) / (n \Sigma x_i^2 - (\Sigma x_i)^2)$
- $a_0 = (\Sigma y_i / n) - a_1 (\Sigma x_i / n)$

Mean of sample y's

Mean of sample x's

Using polyval()

- Returns the result of evaluating a polynomial (represented by a vector of coefficients) at a given x value or vector of x values, i.e.:

  c(1)*x.^degree + c(2)*x.^(degree-1) + ... + c(degree)*x + c(degree+1)

- Usage: y = polyval(c,x)

  c = vector of coefficients

  – e.g., returned from polyfit()

  x = vector of values at which polynomial is to be evaluated

  y = vector of results

Using polyfit()

- Determines a polynomial of given degree that best fits (i.e., minimizes sum of squares of residuals) a given set of x-y points
- Usage:

  c = polyfit(x,y,degree)

  x = vector of values of the independent variable

  y = vector of (measured) values of the dependent variable

  degree = highest power of x in the desired fitted polynomial

  c = vector of coefficients of the polynomial

    length(c) = degree + 1

    The best-fitting polynomial is:

    $c(1)\ x^{degree} + c(2)\ x^{(degree-1)} + \ldots + c(degree)\ x + c(degree+1)$

      i.e., coefficients are ordered from highest power of x to lowest

Curve-fitting with polyfit()

- To fit a line to the data:

  c = polyfit(x,y,1);

  c will be a 2-element vector, defining the equation:

  y = c(1)*x + c(2)

4. Examples

- Measured Data:

x	y
1	2.315
2	0.406
3	-2.373
4	-3.587
5	-5.868

To get a line:

polyfit(x,y,1)

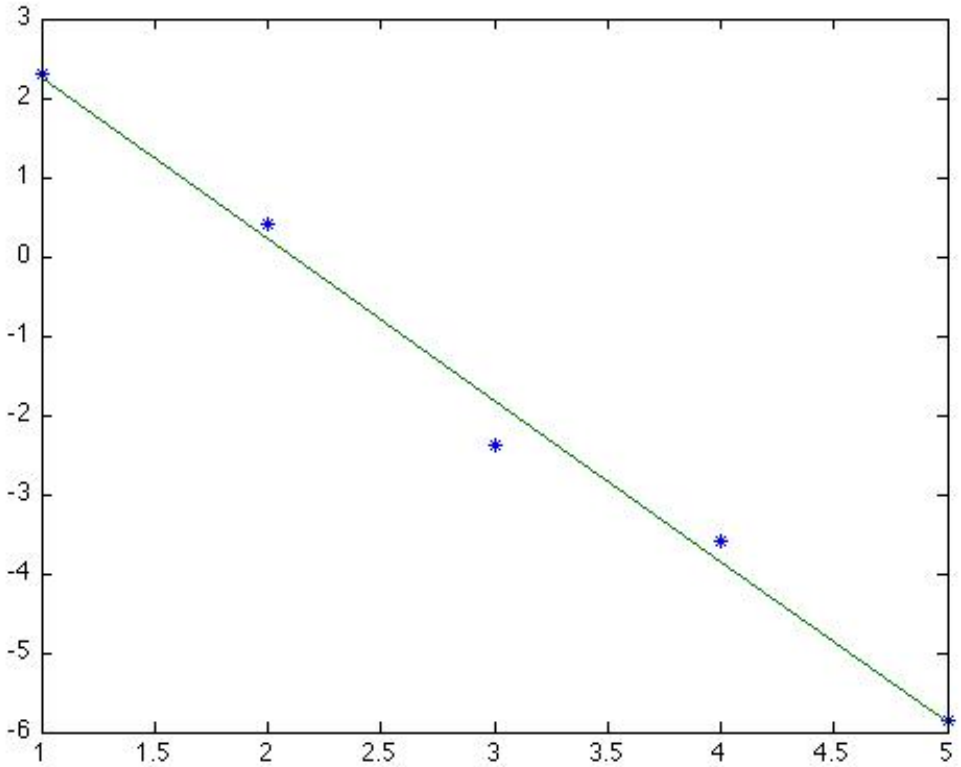
ans = [-2.0359 4.2863]

So the line that produces minimal squares of residuals is

y = -2.0359*x + 4.2863

Example, cont .

- Always plot the curve/line with the data points as a sanity check!
- plot(x, y, '*', x, polyval(c,x))



Example: Hypothesis Testing

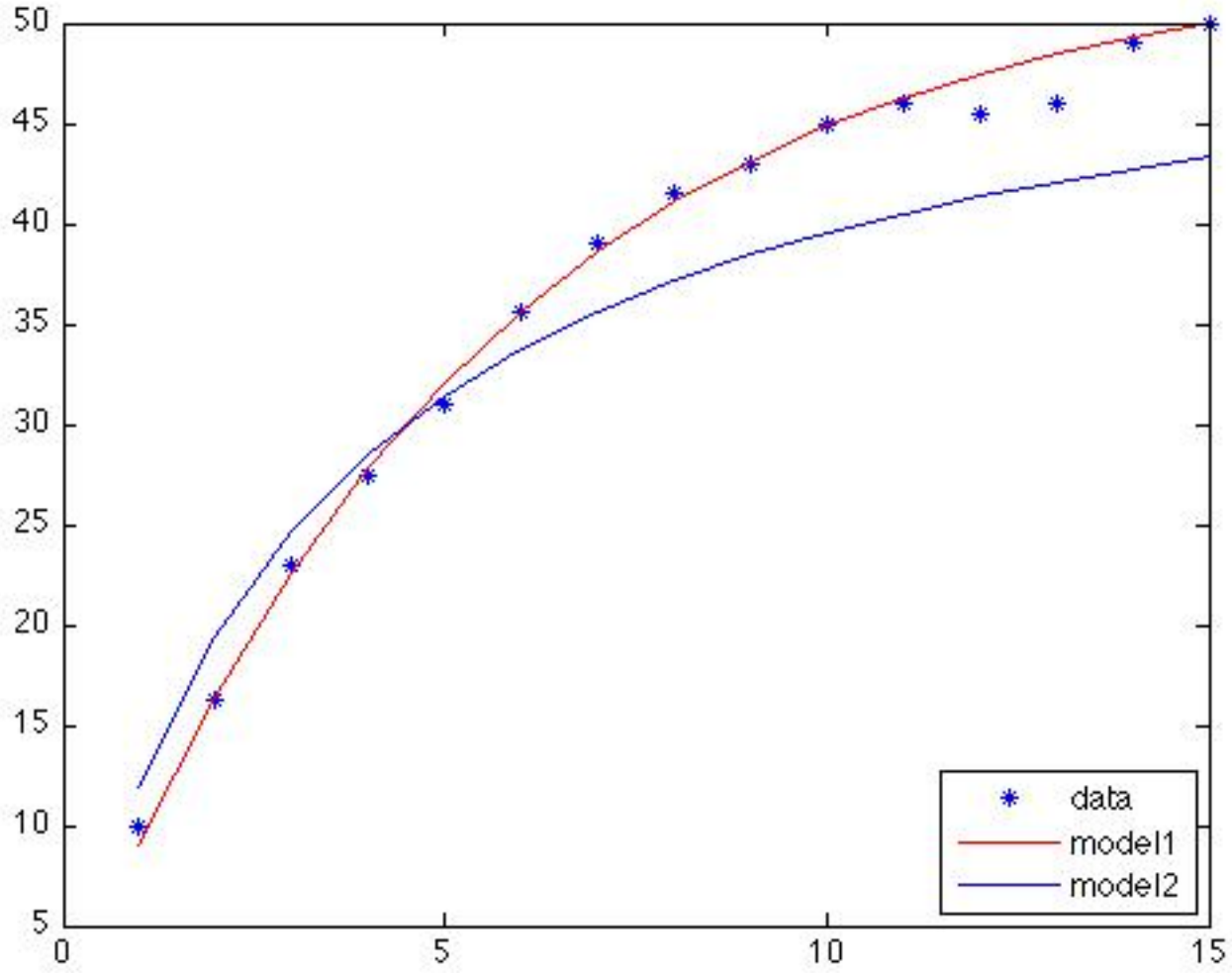
- Velocity of a skydiver vs. time (after jumping)
  - Parameters:

    $g = 9.8\ m/s^2$

    m = mass of skydiver, 68.1 kg

    c = drag coefficient, 12.5 kg/s
  - Model 1: $v(t) = (gm/c)(1 - e^{(-ct/m)})$
  - Model 2: $v(t) = (gm/c)(t/(t+3.5))$

Skydiver: Measured data and models



Which is the better fit?

How can we quantify the fit?

5. Application: Quantifying Fit

- We can use linear regression to compare models!
- Idea: plot the measured y values vs. the values predicted by the model
  - If the model is perfect, the points should all be on the line y = x, i.e. the line with slope 1 and intercept 0
  - So: determine the slope and intercept of the line and compare them to 1 and 0, respectively!

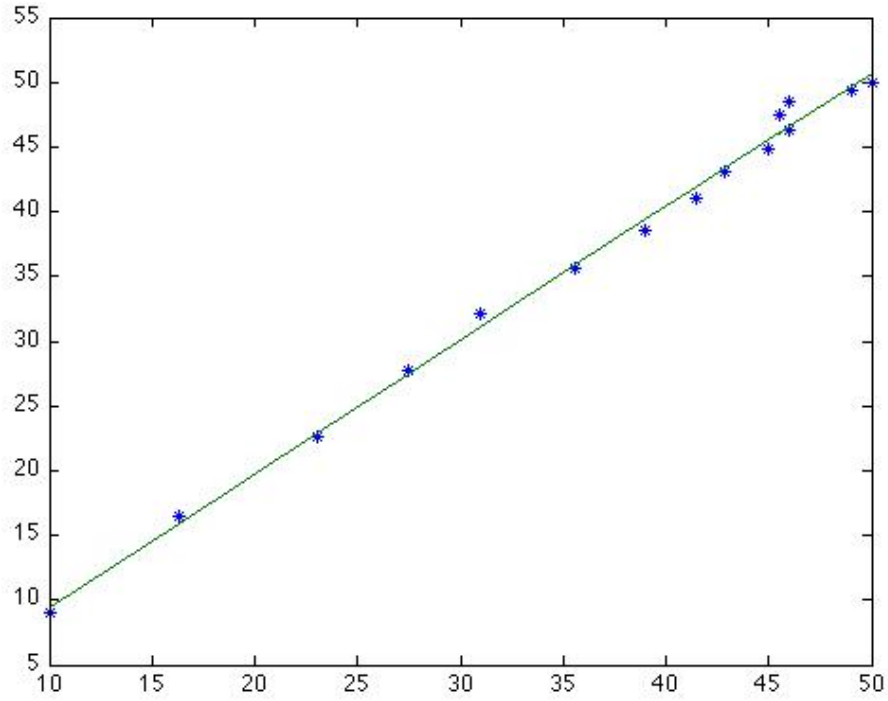
Model 1 fit

- y1 = vector of model-1-predicted values for x = 1:15

  y1 = model1(x)

- c = polyfit(y,y1,1); result: c = [1.0316 -0.8587]

  plot(y,y1,'*',polyval(c,y))



Model 2 fit

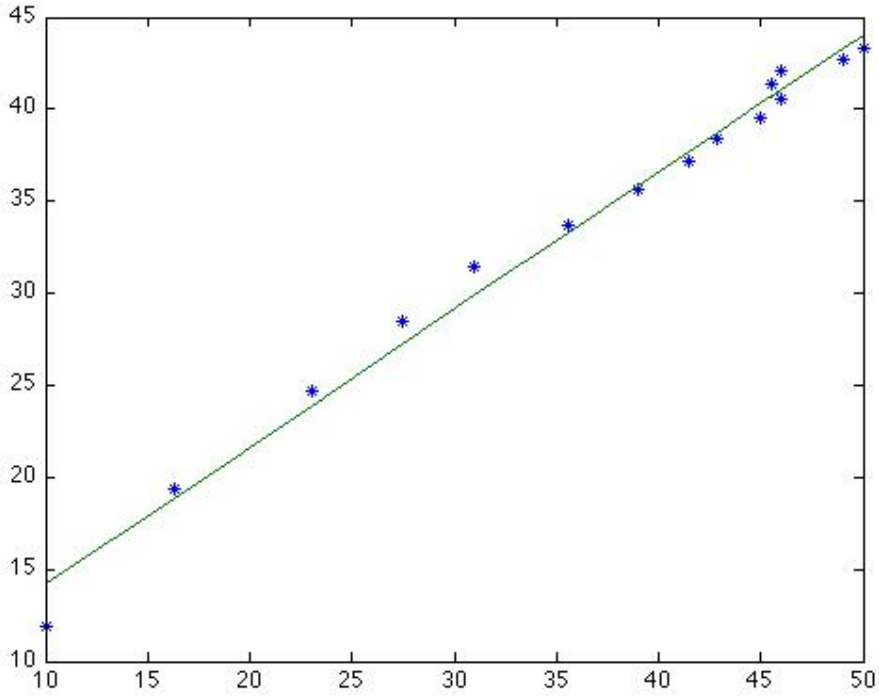
- y2 = model-2-predicted values for x = 1:15

  y2 = model2(x)
- c = polyfit(y,y2,1); result: c = [0.7472 6.6968]

  plot(y,y1,'*',polyval(c,y))

Looking at the coefficients shows that Model 1 is clearly better.

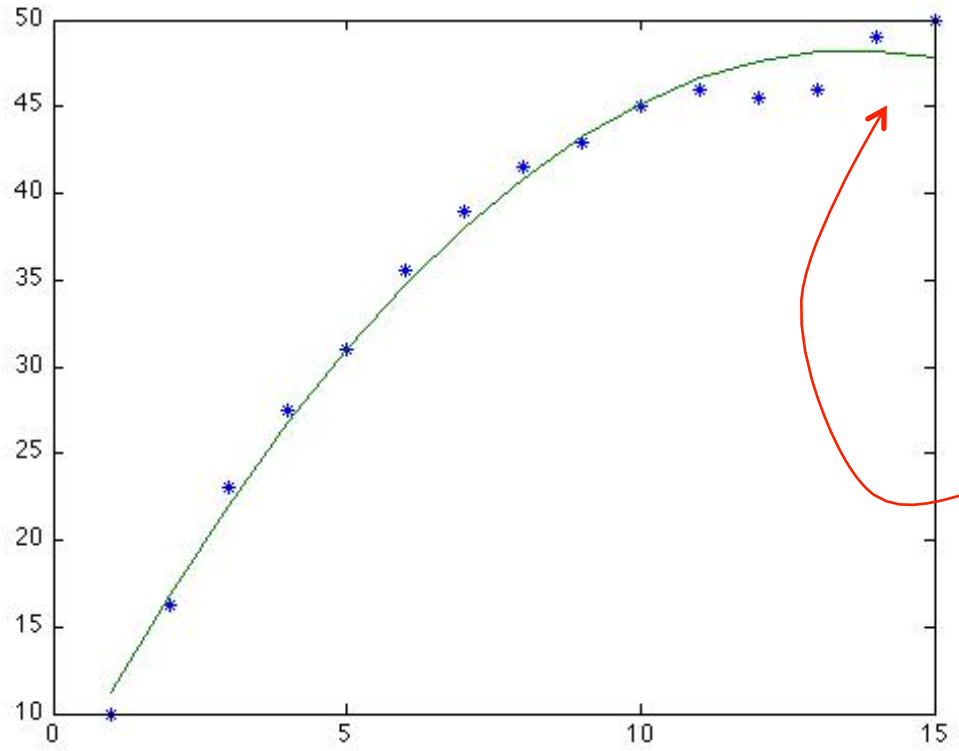
But... be careful!



Fitting a parabola

Fit a parabola (degree-2 polynomial) to the data:

c = polyfit(x,y,2); plot(x,y, '*', x, polyval(c,x))



The slope of the "noise line" for this model is 0.99, and the intercept is 0.35.

But it is not a good model, because the velocity decreases for larger t!

The Moral of the Story

- Do not blindly rely on any quantitative metric.
- Always use your understanding of the application – in this case, that the velocity should not decrease (until the skydiver hits the ground) – to sanity-check what the numbers are telling you.

6. Using Linear Regression to Solve Nonlinear Relationships

- General Procedure:
  - Transform the nonlinear relationship into a linear one
  - Transform the measured data as needed
  - Solve for coefficients using polyfit()
  - Transform coefficients back to fit original formula

Nonlinear Example 1

- $y = \alpha e^{\beta x}$

  take natural log of both sides: $\ln y = \ln \alpha + \beta x$

  This is linear – so take log of measured y's, use polyfit() to get β and ln α, compute α = exp(ln α)

  1. logy = log(y)
  2. c = polyfit(x,logy,1)
  3. beta = c(1);
  4. alpha = exp(c(2));
  5. plot(x,y, '*', x, alpha*exp(beta*x)) % compare!

Nonlinear Example 2

- $y = \alpha x^{\beta}$

  Note: not polynomial – β unknown, not necessarily integer

- Again, take log of both sides:

  $\log y = \log \alpha + \beta * \log x$

  This time have to take log of <u>both</u> x's and measured y's:

  1. logy = log(y)
  2. logx = log(x)
  3. c = polyfit(logx,logy,1)
  4. beta = c(1);
  5. alpha = exp(c(2));
  6. plot(x,y, '*', x, alpha*(x.^beta)) % compare!

Nonlinear Example 3

- $y = \alpha x/(x+\beta)$
- Take reciprocal of both sides:

$$1/y = (x+\beta)/(\alpha x) = 1/\alpha + (\beta/\alpha)*(1/x)$$

Take reciprocal of x's and measured y's, solve for $1/\alpha$ and β/α

1. ry =1/y
2. rx = 1/x
3. c = polyfit(rx,ry,1)
4. alpha = 1/c(2);
5. beta = c(1)*alpha;
6. plot(x,y, '*', x, (alpha*x)./(x + beta)) % compare!

Note ./ form!

Multiple Linear Regression

- When y is a function of multiple x's:

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n$$

- As before:
  - Residual = difference between predicted and observed/ measured values
  - Want to minimize sum of squares of residuals:

$$S_r = \Sigma\, e_i^2 = \Sigma(y_i - a_0 - a_1x_{1i} - \ldots - a_nx_{ni})^2$$

"Predicted" value for x_i

- As with polynomial regression, take partial derivatives with respect to each a_i (e.g., for two x's):

$$\delta S_r/\delta a_0 = -2\,\Sigma\,(y_i - a_0 - a_1x_{1i} - a_2x_{2i})$$

$$\delta S_r/\delta a_1 = -2\,\Sigma\,x_{1i}\,[(y_i - a_0 - a_1x_{1i} - a_2x_{2i})]$$

$$\delta S_r/\delta a_1 = -2\,\Sigma\,x_{2i}\,[(y_i - a_0 - a_1x_{1i} - a_2x_{2i})]$$

Multiple Linear Regression

- Set partial derivatives equal to 0, rearrange to get normal (matrix) equation:

$$\begin{bmatrix} n & \Sigma x_{1i} & \Sigma x_{2i} \\ \Sigma x_{1i} & \Sigma x_{1i}^2 & \Sigma x_{1i}x_{2i} \\ \Sigma x_{2i} & \Sigma x_{1i}x_{2i} & \Sigma x_{2i}^2 \end{bmatrix} \begin{bmatrix} a0 \\ a1 \\ a2 \end{bmatrix} = \begin{bmatrix} \Sigma y_i \\ \Sigma x_{1i}y_i \\ \Sigma x_{2i}y_i \end{bmatrix}$$

(all Σ's are over i)

(note symmetry in x_{1i} and x_{2i})

- Solve this to get a_i's!

6. A Quick Word on File I/O

- You can send output to files via fprintf().
- Basic model (output):
  - "Open" the file with fopen()
    - Give it the filename as argument (note: path)
    - fopen() returns a "handle" (integer)
  - Write to the file with fprintf()
    - Give it the handle as first argument

Sending Output to a File

```
fhandle = fopen('myOutput.txt','w');
fprintf(fhandle,'This output goes into the file!\n');
x = ....  % calculations
y = ....  % more calculations
fprintf(fhandle,'x = %10.2f, y = %10.4f.\n',x,y);
```

Reading Input from a File

- You can read input from a file
- fhandle = fopen(<filename>,'r'); % r = read
- A = fscanf(fhandle,<formatstring>)
- Returns elements in A in column order

  Example:

  A = fscanf(fhandle,'%d\n') % reads one integer/line