# Data Modeling and Least Squares Fitting 2

COS 323

# Last time

- Data modeling
- Motivation of least-squares error
- Formulation of ***linear*** least-squares model:

$$y_i = af(\vec{x}_i) + bg(\vec{x}_i) + ch(\vec{x}_i) + \cdots$$
$$\text{Given } (\vec{x}_i, y_i), \text{ solve for } a, b, c, \ldots$$

- Solving using normal equations, pseudoinverse
- Illustrating least-squares with special cases: constant, line
- Weighted least squares
- Evaluating model quality

# Nonlinear Least Squares

- Some problems can be rewritten to linear

$$y = ae^{bx}$$

$$\Rightarrow (\log y) = (\log a) + bx$$

- Fit data points $(x_i, \log y_i)$ to $a^* + bx$, $a = e^{a^*}$
- Big problem: this no longer minimizes squared error!

# Nonlinear Least Squares

- Can write error function, minimize directly

$$\chi^2 = \sum_i \left(y_i - f(x_i, a, b, \ldots)\right)^2$$

$$\text{Set } \frac{\partial}{\partial a} = 0, \; \frac{\partial}{\partial b} = 0, \text{ etc.}$$

- For the exponential, no analytic solution for a, b:

$$\chi^2 = \sum_i \left(y_i - ae^{bx_i}\right)^2$$

$$\frac{\partial}{\partial a} = \sum_i -2e^{bx_i}\left(y_i - ae^{bx_i}\right) = 0$$

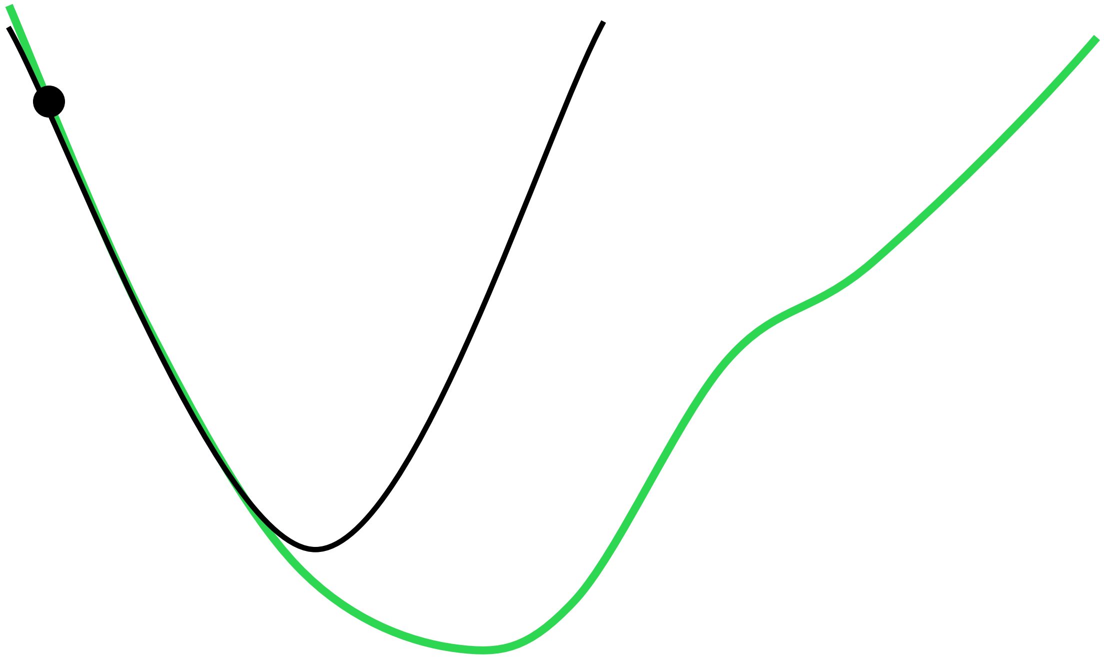
$$\frac{\partial}{\partial b} = \sum_i -2ax_i e^{bx_i}\left(y_i - ae^{bx_i}\right) = 0$$

# Newton's Method

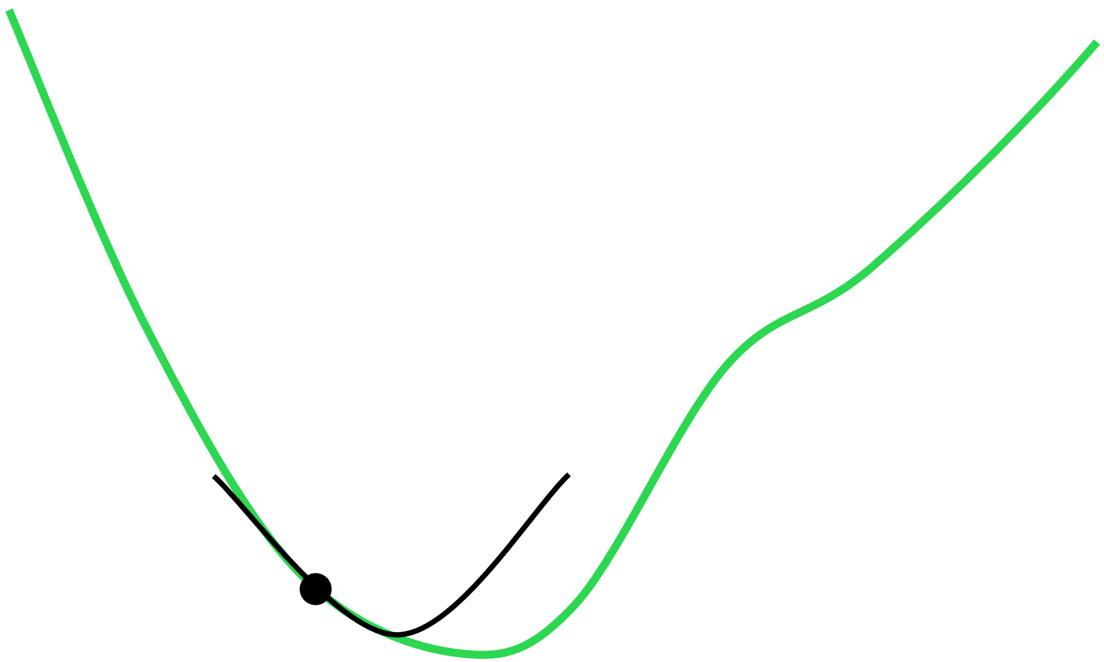
- Apply Newton's method for minimization:
  - 1-dimensional: $$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$
  - n-dimensional: $$\begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_{i+1} = \begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_i - H^{-1}G$$

  where H is Hessian (matrix of all 2nd derivatives) and G is gradient (vector of all 1st derivatives)

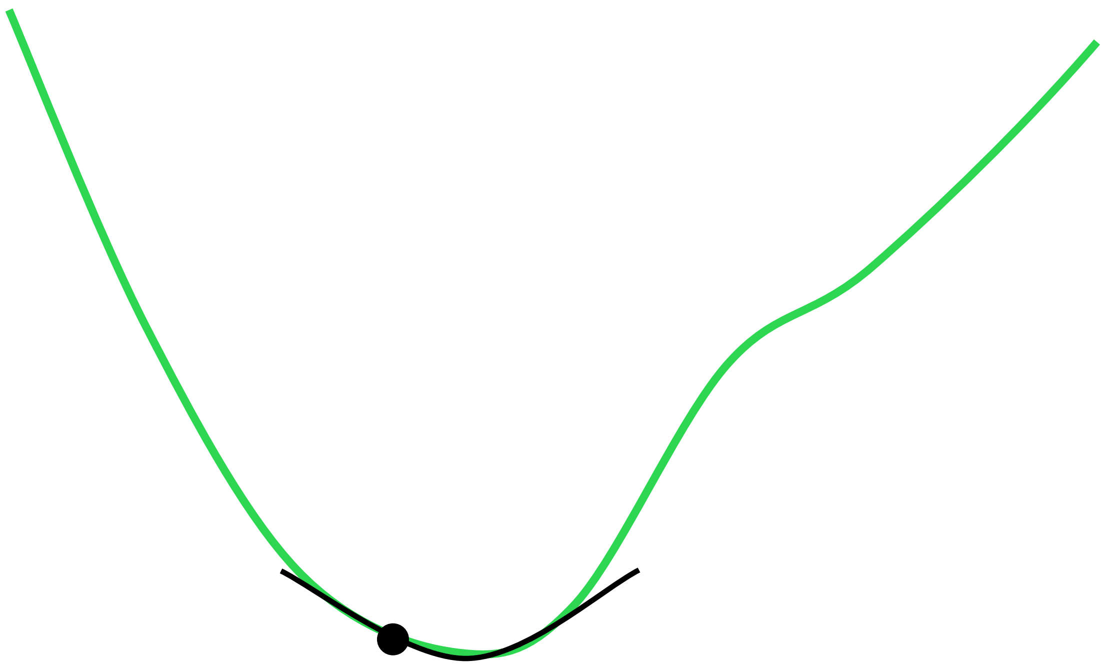
# Newton's Method



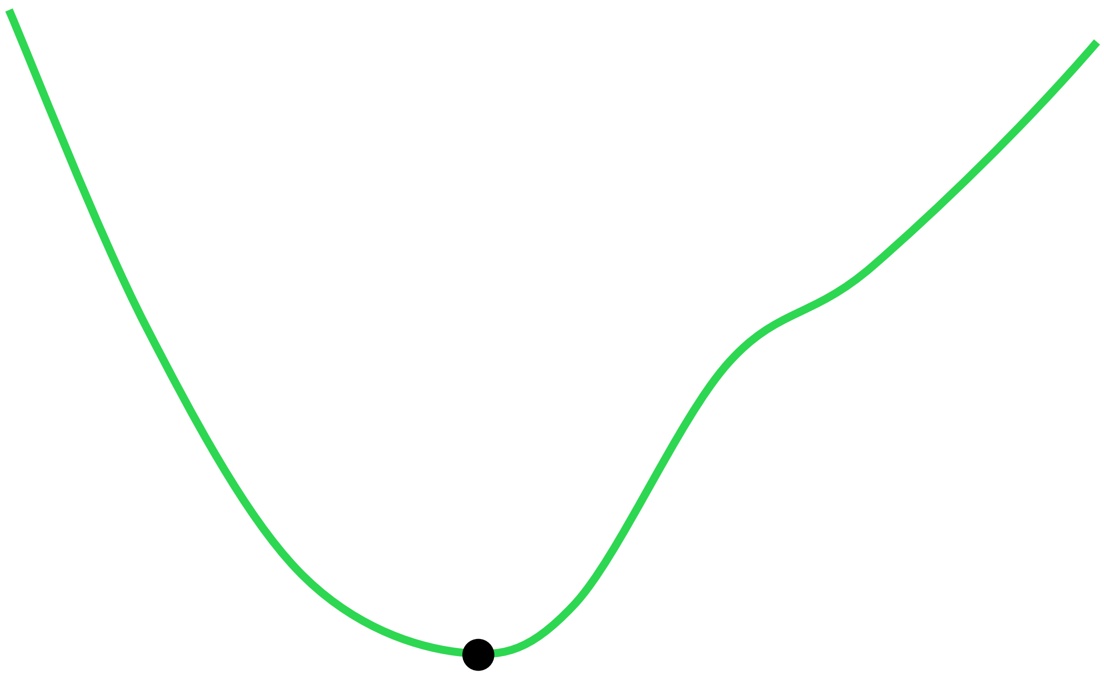
# Newton's Method



# Newton's Method



# Newton's Method



# Newton's Method

- Apply Newton's method for minimization:
  - 1-dimensional: $$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$
  - n-dimensional: $$\begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_{i+1} = \begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_i - \mathbf{H}^{-1}\mathbf{G}$$

where H is Hessian (matrix of all 2$^{nd}$ derivatives) and G is gradient (vector of all 1$^{st}$ derivatives)

# Newton's Method for Least Squares

$$\chi^2(a,b,\ldots) = \sum_i \left(y_i - f(x_i,a,b,\ldots)\right)^2$$

$$\mathbf{G} = \begin{bmatrix} \frac{\partial(\chi^2)}{\partial a} \\ \frac{\partial(\chi^2)}{\partial b} \\ \vdots \end{bmatrix} = \begin{bmatrix} \sum_i -2\frac{\partial f}{\partial a}\left(y_i - f(x_i,a,b,\ldots)\right) \\ \sum_i -2\frac{\partial f}{\partial b}\left(y_i - f(x_i,a,b,\ldots)\right) \\ \vdots \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2(\chi^2)}{\partial a^2} & \frac{\partial^2(\chi^2)}{\partial a \partial b} & \cdots \\ \frac{\partial^2(\chi^2)}{\partial a \partial b} & \frac{\partial^2(\chi^2)}{\partial b^2} & \cdots \\ & \vdots & \end{bmatrix}$$

- Gradient has 1st derivatives of $f$, Hessian 2nd

# Gauss-Newton Iteration

- Consider 1 term of Hessian:

$$\frac{\partial^2(\chi^2)}{\partial a^2} = \frac{\partial}{\partial a}\left(\sum_i -2\,\tfrac{\partial f}{\partial a}\big(y_i - f(x_i,a,b,\ldots)\big)\right)$$

$$= -2\sum_i \tfrac{\partial^2 f}{\partial a^2}\cancel{\big(y_i - f(x_i,a,b,\ldots)\big)} + 2\sum_i \tfrac{\partial f}{\partial a}\tfrac{\partial f}{\partial a}$$

- If close to answer, residual is close to 0,
so ignore it → eliminates need for 2nd derivatives

# Gauss-Newton Iteration

- Consider 1 term of Hessian:

$$\frac{\partial^2(\chi^2)}{\partial a^2} = \frac{\partial}{\partial a}\left(\sum_i -2\tfrac{\partial f}{\partial a}\big(y_i - f(x_i,a,b,\ldots)\big)\right)$$

$$= -2\sum_i \tfrac{\partial^2 f}{\partial a^2}\cancel{\big(y_i - f(x_i,a,b,\ldots)\big)} + 2\sum_i \tfrac{\partial f}{\partial a}\tfrac{\partial f}{\partial a}$$

- The Gauss-Newton method approximates

$$\mathbf{H} \approx 2\mathbf{J}^{\mathrm{T}}\mathbf{J}$$

(Only for least-squares!)

# Gauss-Newton Iteration

$$\begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_{i+1} = \begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_i + s_i$$

To find Gauss – Newton update $s_i = -\mathbf{H}^{-1}\mathbf{G}$, solve

$$\mathrm{J}_i^{\mathrm{T}}\mathrm{J}_i s_i = \mathrm{J}_i^{\mathrm{T}} r_i$$

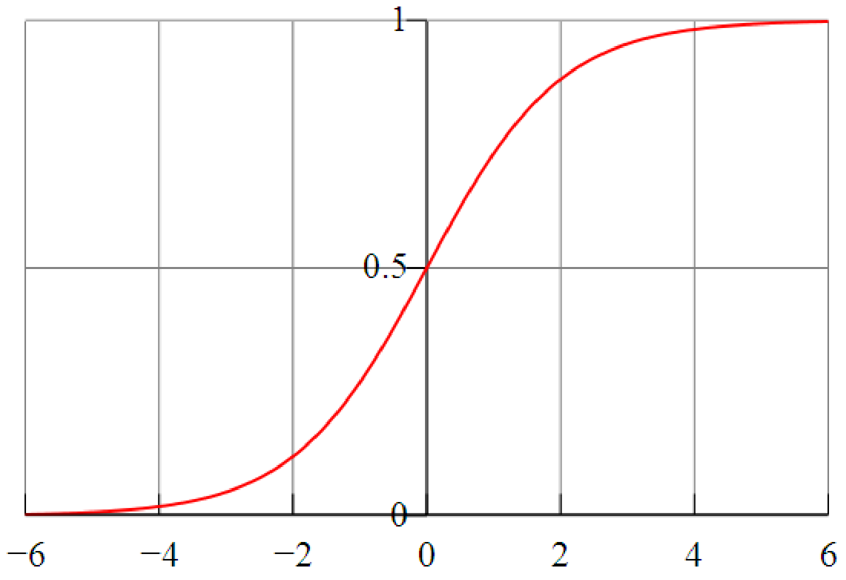
$$J = \begin{pmatrix} \frac{\partial f}{\partial a}(x_1) & \frac{\partial f}{\partial b}(x_1) & \cdots \\ \frac{\partial f}{\partial a}(x_2) & \frac{\partial f}{\partial b}(x_2) & \\ \vdots & & \ddots \end{pmatrix}, r = \begin{pmatrix} y_1 - f(x_1, a, b, \cdots) \\ y_2 - f(x_2, a, b, \cdots) \\ \vdots \end{pmatrix}$$

# Example: Logistic Regression

- Model probability of an event based on values of explanatory variables, using generalized linear model, logistic function g(z)

$$p(\vec{x}) = g(ax_1 + bx_2 + \cdots)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



# Logistic Regression

- Assumes positive and negative examples are normally distributed, with different means but same variance
- Applications: predict odds of election victories, sports events, medical outcomes, etc.
- Estimate parameters a, b, … using Gauss-Newton on individual positive, negative examples
- Handy hint: $g'(z) = g(z)\ (1-g(z))$

# Gauss-Newton++:
# The Levenberg-Marquardt Algorithm

# Levenberg-Marquardt

- Newton (and Gauss-Newton) work well when close to answer, terribly when far away
- Steepest descent safe when far away
- Levenberg-Marquardt idea: let's do both

$$\begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_{i+1} = \begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_{i} - \alpha \mathbf{G} - \beta \begin{pmatrix} \sum \frac{\partial f}{\partial a}\frac{\partial f}{\partial a} & \sum \frac{\partial f}{\partial a}\frac{\partial f}{\partial b} & \cdots \\ \sum \frac{\partial f}{\partial a}\frac{\partial f}{\partial b} & \sum \frac{\partial f}{\partial b}\frac{\partial f}{\partial b} & \cdots \\ & \vdots & \end{pmatrix}^{-1} \mathbf{G}$$

Steepest descent

Gauss-Newton

# Levenberg-Marquardt

- Trade off between constants depending on how far away you are…
- Clever way of doing this:

$$\begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_{i+1} = \begin{pmatrix} a \\ b \\ \vdots \end{pmatrix}_{i} - \begin{pmatrix} (1+\lambda)\Sigma \frac{\partial f}{\partial a}\frac{\partial f}{\partial a} & \Sigma \frac{\partial f}{\partial a}\frac{\partial f}{\partial b} & \cdots \\ \Sigma \frac{\partial f}{\partial a}\frac{\partial f}{\partial b} & (1+\lambda)\Sigma \frac{\partial f}{\partial b}\frac{\partial f}{\partial b} & \cdots \\ & \vdots & \end{pmatrix}^{-1} \mathbf{G}$$

- If $\lambda$ is small, mostly like Gauss-Newton
- If $\lambda$ is big, matrix becomes mostly diagonal, behaves like steepest descent

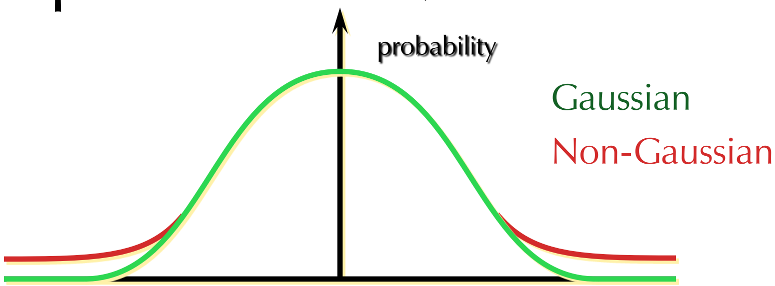
# Levenberg-Marquardt

- Final bit of cleverness: adjust $\lambda$ depending on how well we're doing
  - Start with some $\lambda$, e.g. 0.001
  - If last iteration *decreased* error, *accept* the step and *decrease* $\lambda$ to $\lambda/10$
  - If last iteration *increased* error, *reject* the step and *increase* $\lambda$ to $10\lambda$
- Result: fairly stable algorithm, not too painful (no 2$^{nd}$ derivatives), used a lot

# Dealing with Outliers

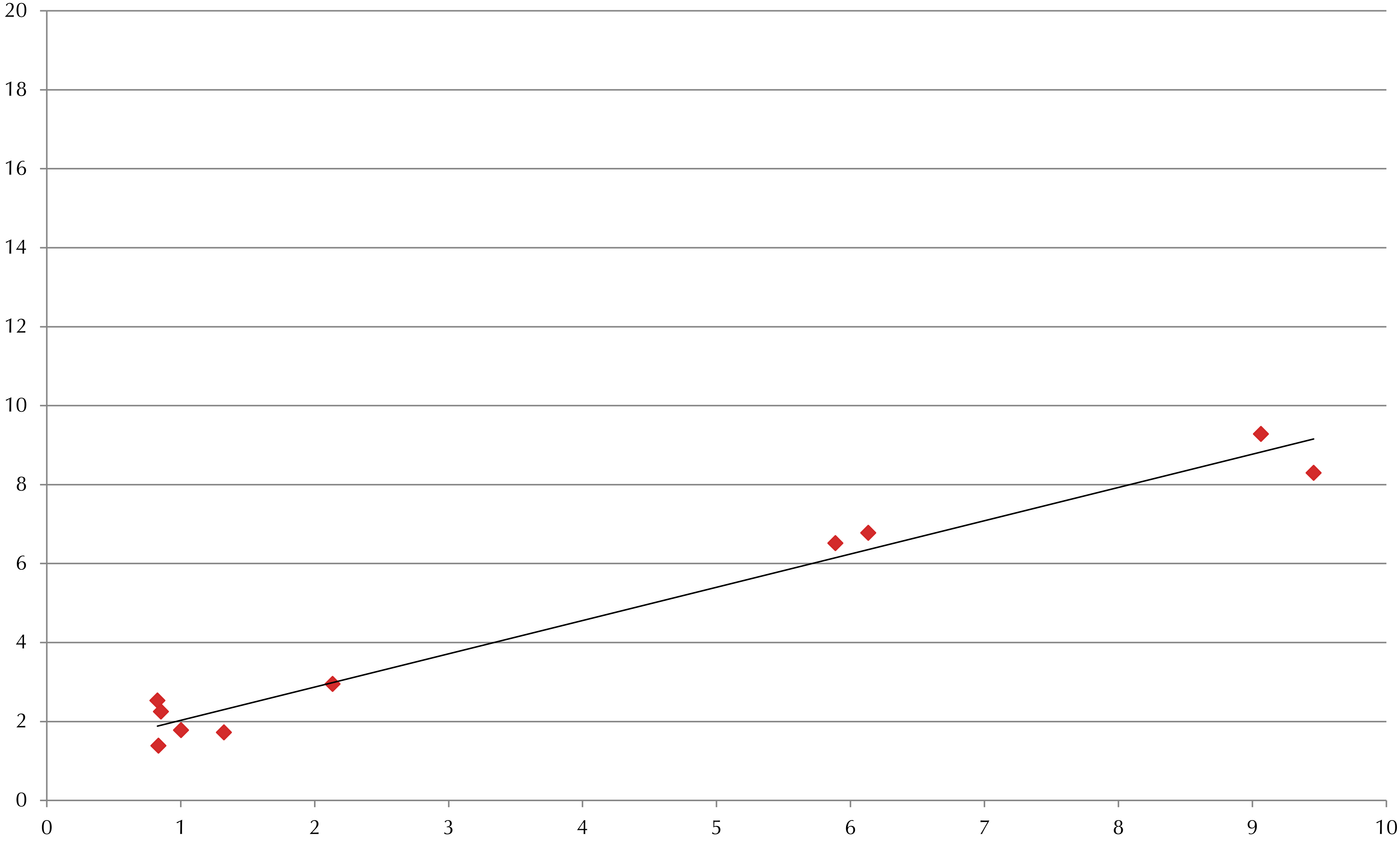
# Outliers

- A lot of derivations assume Gaussian distribution for errors
- Unfortunately, nature (and experimenters) sometimes don’t cooperate

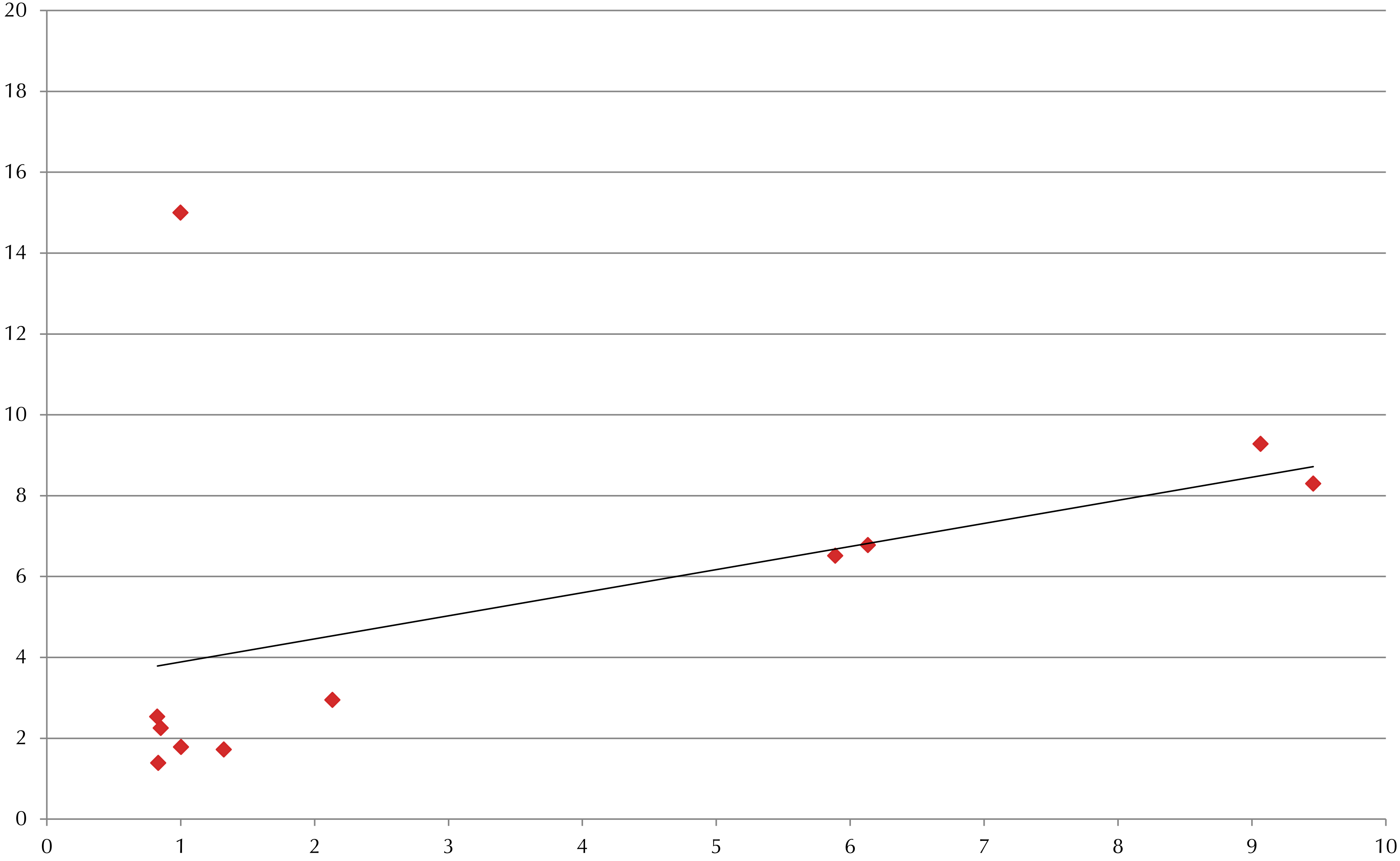



- Outliers: points with extremely low probability of occurrence (according to Gaussian statistics)
- Can have strong influence on least squares

# Example: without outlier



# Example: with outlier



# Robust Estimation

- Goal: develop parameter estimation methods insensitive to *small* numbers of *large* errors
- General approach: try to give large deviations less weight
- e.g., **Median** is a robust measure, **mean** is not
- **M-estimators: minimize some function other than square of y – f(x,a,b,…)**

# Least Absolute Value Fitting

- Minimize $\sum_i \left| y_i - f(x_i, a, b, \ldots) \right|$
  instead of $\sum_i \left( y_i - f(x_i, a, b, \ldots) \right)^2$
- Points far away from trend get comparatively less influence

# Example: Constant

- For constant function $y = a$, minimizing $\Sigma(y-a)^2$ gave $a = \text{mean}$
- Minimizing $\Sigma|y-a|$ gives $a = \text{median}$

# Least Squares vs. Least Absolute Deviations

- LS:
  - Not robust
  - Stable, unique solution
  - Solve with normal equations, Gauss-Newton, etc.
- LAD
  - Robust
  - Unstable, not necessarily unique
  - Nasty function (discontinuous derivative): requires iterative solution method (e.g. simplex)

# Iteratively Reweighted Least Squares

- Sometimes-used approximation:
  convert to iteratively weighted least squares

$$\sum_i \left| y_i - f(x_i, a, b, \ldots) \right|$$

$$= \sum_i \frac{1}{\left| y_i - f(x_i, a, b, \ldots) \right|} \left( y_i - f(x_i, a, b, \ldots) \right)^2$$

$$= \sum_i w_i \left( y_i - f(x_i, a, b, \ldots) \right)^2$$

with $w_i$ based on previous iteration

# Review: Weighted Least Squares

- Define weight matrix W as

$$\mathbf{W} = \begin{pmatrix} w_1 & & & & 0 \\ & w_2 & & & \\ & & w_3 & & \\ & & & w_4 & \\ 0 & & & & \ddots \end{pmatrix}$$

- Then solve weighted least squares via

$$\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{A}\,x = \mathbf{A}^{\mathrm{T}}\mathbf{W}\,b$$

# M-Estimators

Different options for weights

- Give even less weight to outliers

$$w_i = \frac{1}{\left|y_i - f(x_i, a, b, \ldots)\right|}$$ L$_1$

$$w_i = \frac{1}{\varepsilon + \left|y_i - f(x_i, a, b, \ldots)\right|}$$ "Fair"

$$w_i = \frac{1}{\varepsilon + \left(y_i - f(x_i, a, b, \ldots)\right)^2}$$ Cauchy / Lorentzian

$$w_i = e^{-k\left(y_i - f(x_i, a, b, \ldots)\right)^2}$$ Welsch

# Iteratively Reweighted Least Squares

- Danger!  This is not guaranteed to converge to the right answer!
  - Needs good starting point, which is available if initial least squares estimator is reasonable
  - In general, works OK if few outliers, not too far off

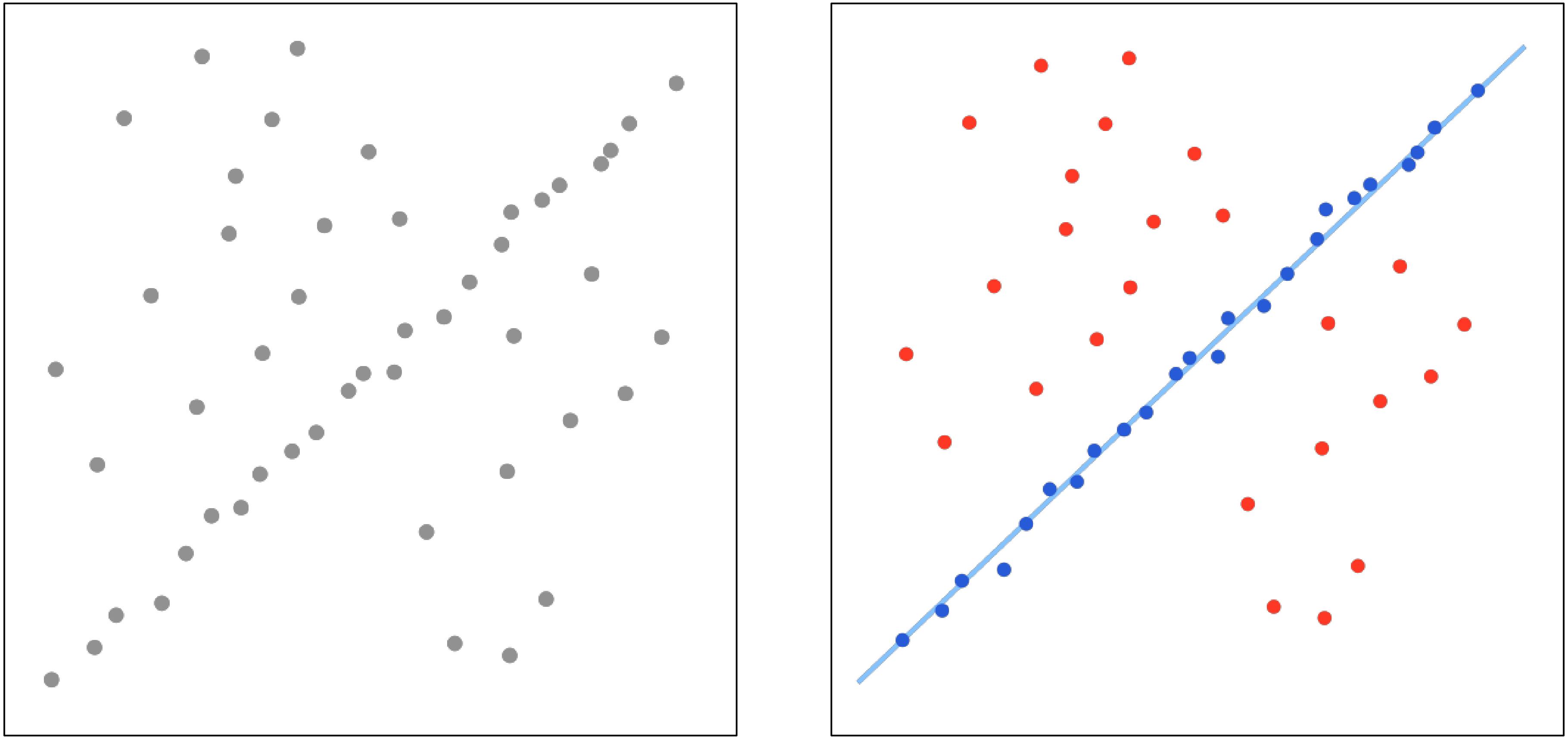
# Outlier Detection and Rejection

- Special case of IRWLS: set weight = 0 if outlier, 1 otherwise
- Detecting outliers: $(y_i - f(x_i))^2 >$ threshold
  - One choice: multiple of mean squared difference
  - Better choice: multiple of *median* squared difference
  - Can iterate…
  - As before, not guaranteed to do anything reasonable, tends to work OK if only a few outliers

# RANSAC

- RANdom SAmple Consensus: desgined for bad data (in best case, up to 50% outliers)
- Take many *minimal* random subsets of data
  - Compute fit for each sample
  - See how many points agree: $(y_i - f(x_i))^2 < \text{threshold}$
  - Threshold user-specified or estimated from more trials
- At end, use fit that agreed with most points
  - Can do one final least squares with all inliers

# RANSAC



# Least Squares in Practice

# Least Squares in Practice

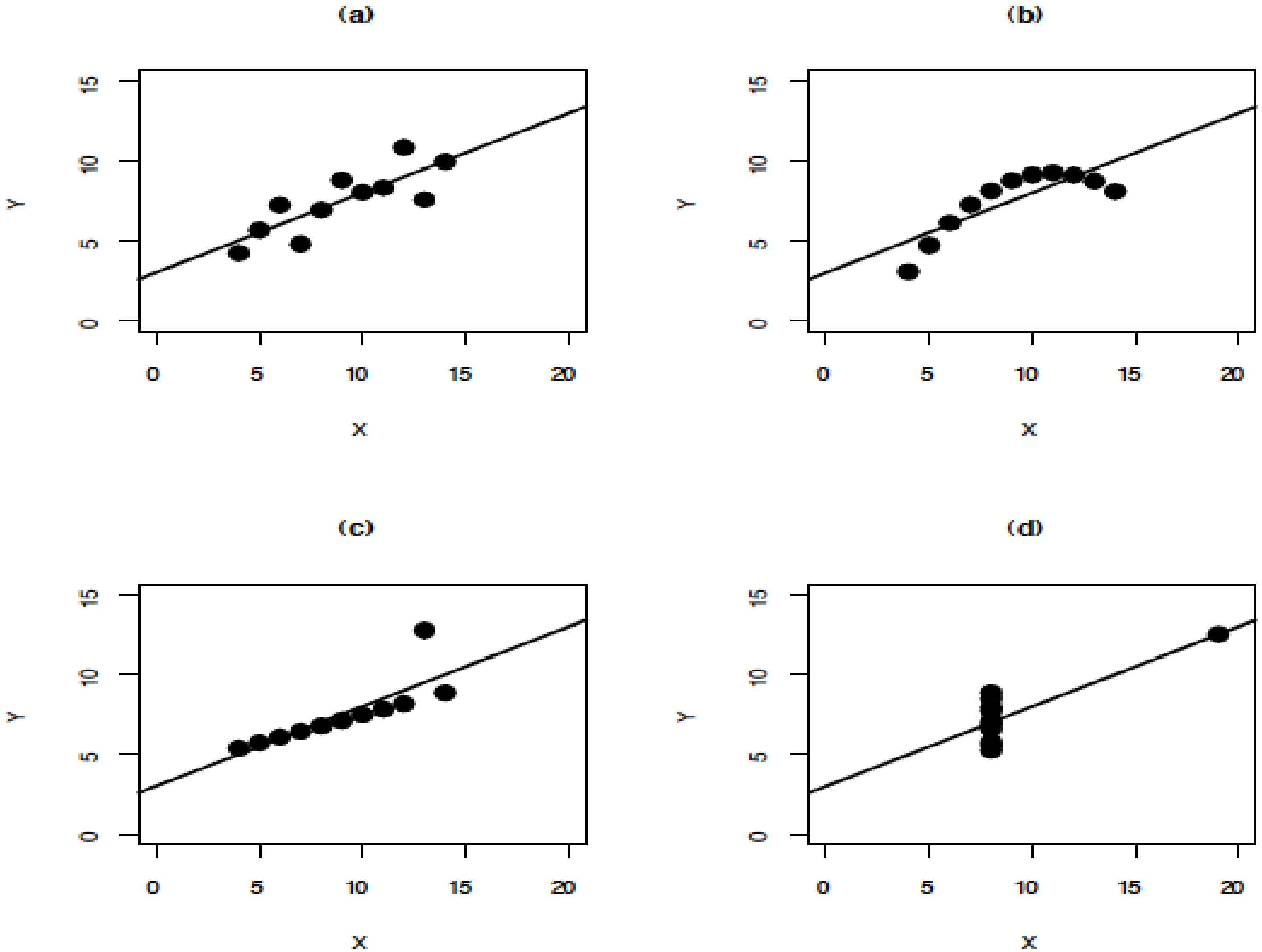
- More data is better $\sigma^2 = \frac{\chi^2}{n-m}\mathbf{C}$
  – uncertainty in estimated parameters goes down slowly: like 1/sqrt(# samples)
- Good correlation **doesn't mean a model is good**
  – use visualizations and reasoning, too.

# Anscombe's Quartet

| Dataset 1 | | Dataset 2 | | Dataset 3 | | Dataset 4 | |
|---|---|---|---|---|---|---|---|
| $x$ | $y$ | $x$ | $y$ | $x$ | $y$ | $x$ | $y$ |
| 10 | 8.04 | 10 | 9.14 | 10 | 7.46 | 8 | 6.58 |
| 8 | 6.95 | 8 | 8.14 | 8 | 6.77 | 8 | 5.76 |
| 13 | 7.58 | 13 | 8.74 | 13 | 12.74 | 8 | 7.71 |
| 9 | 8.81 | 9 | 8.77 | 9 | 7.11 | 8 | 8.84 |
| 11 | 8.33 | 11 | 9.26 | 11 | 7.81 | 8 | 8.47 |
| 14 | 9.96 | 14 | 8.10 | 14 | 8.84 | 8 | 7.04 |
| 6 | 7.24 | 6 | 6.13 | 6 | 6.08 | 8 | 5.25 |
| 4 | 4.26 | 4 | 3.10 | 4 | 5.39 | 19 | 12.50 |
| 12 | 10.84 | 12 | 9.13 | 12 | 8.15 | 8 | 5.56 |
| 7 | 4.82 | 7 | 7.26 | 7 | 6.42 | 8 | 7.91 |
| 5 | 5.68 | 5 | 4.74 | 5 | 5.73 | 8 | 6.89 |

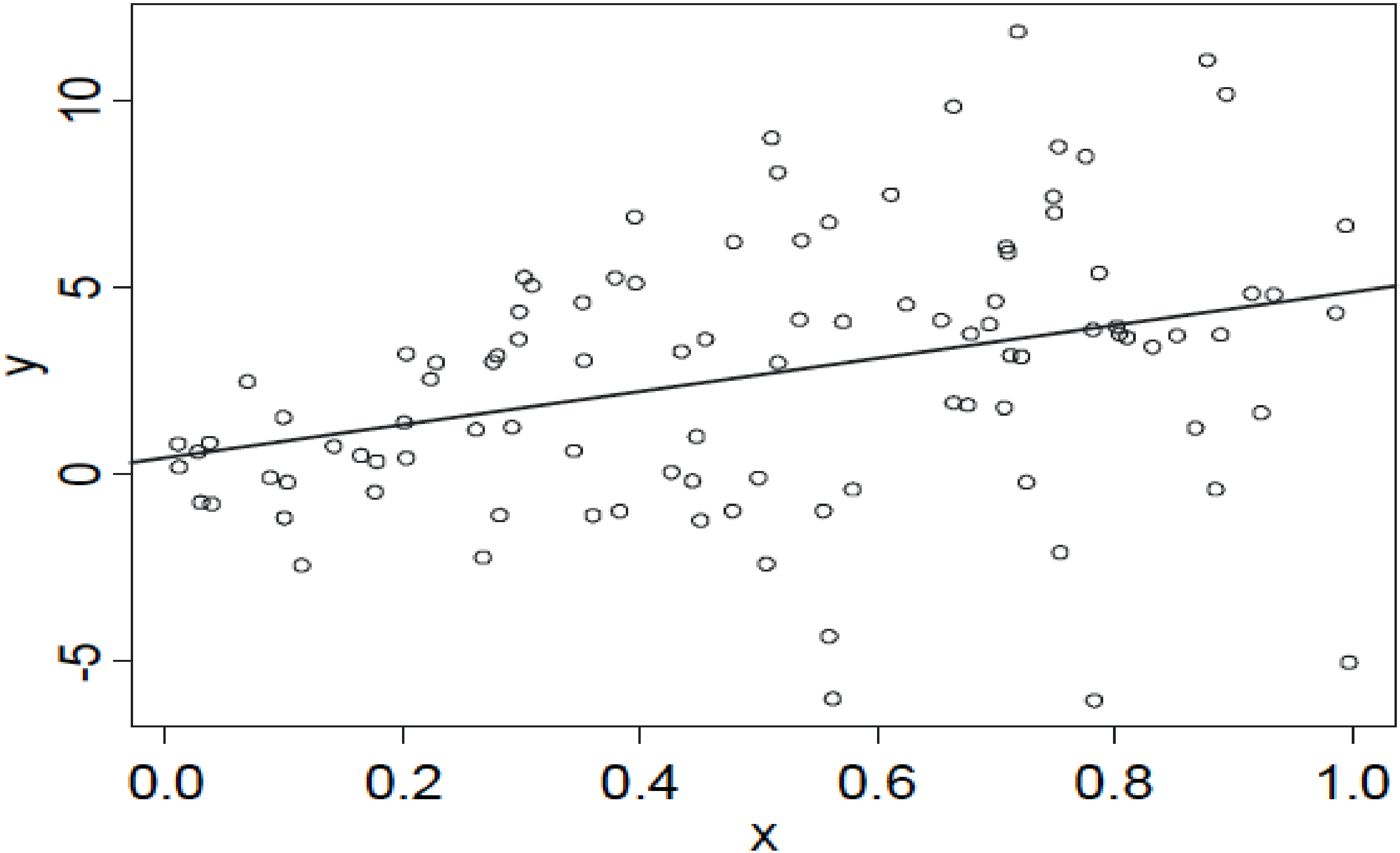
$$y = 3.0 + 0.5x$$
$$r = 0.82$$

# Anscombe's Quartet



# Least Squares in Practice

- More data is better
- Good correlation **doesn’t mean a model is good**
- Many circumstances call for (slightly) more sophisticated models than least squares
  - Generalized linear models, regularized models (e.g., LASSO), PCA, …

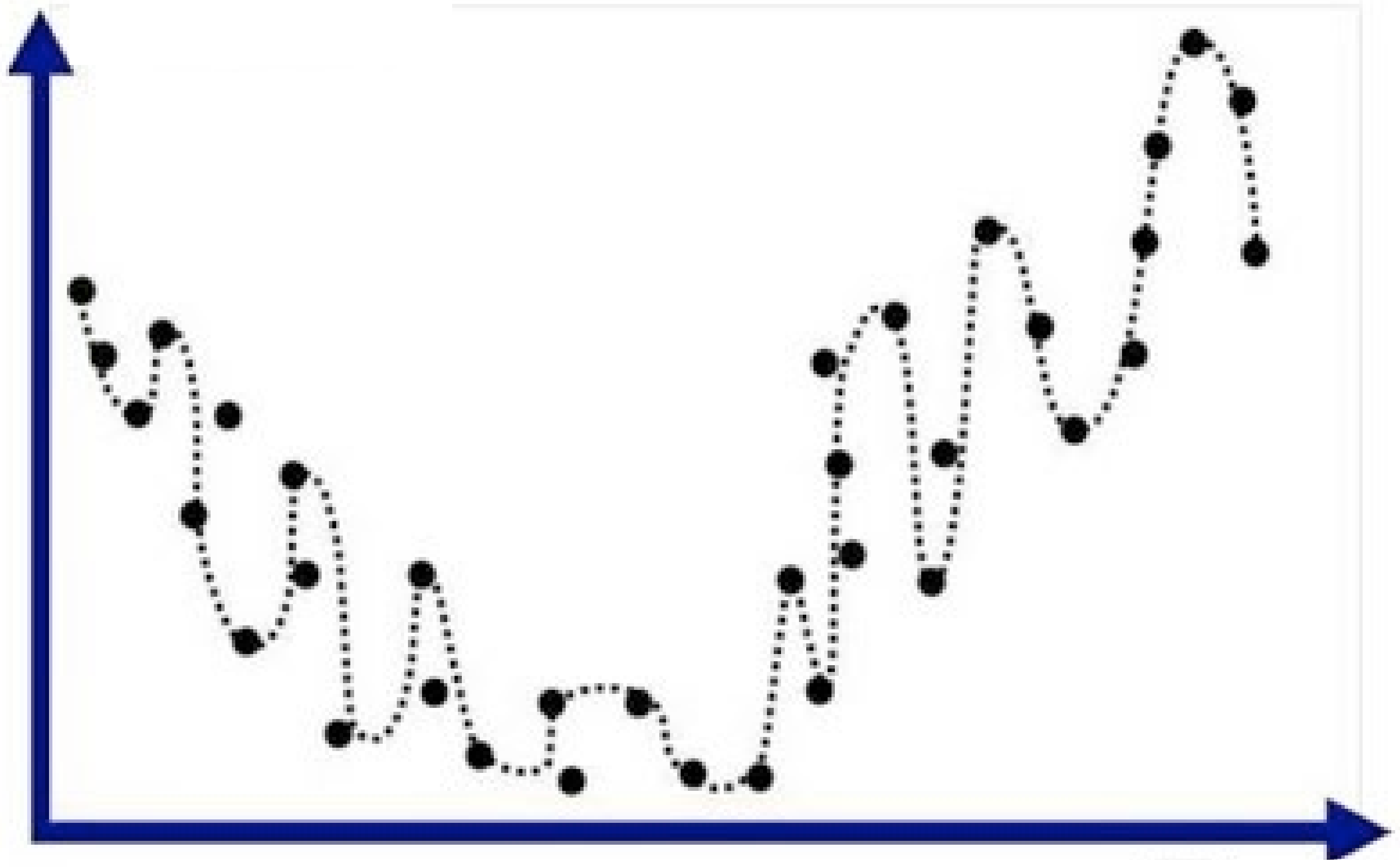



Residuals depend on x (heteroscedastic):
Assumptions of linear least squares not met

# Least Squares in Practice

- More data is better
- Good correlation **doesn't mean a model is good**
- Many circumstances call for (slightly) more sophisticated models than linear LS
- Sometimes a model's fit can be **too good** ("overfitting")
  - more parameters may make it easier to overfit

# Overfitting



# Least Squares in Practice

- More data is better
- Good correlation **doesn't mean a model is good**
- Many circumstances call for (slightly) more sophisticated models than linear LS
- Sometimes a model's fit can be **too good**
- All of these minimize "vertical" squared distance
  - Square, vertical distance not always appropriate