# Recap

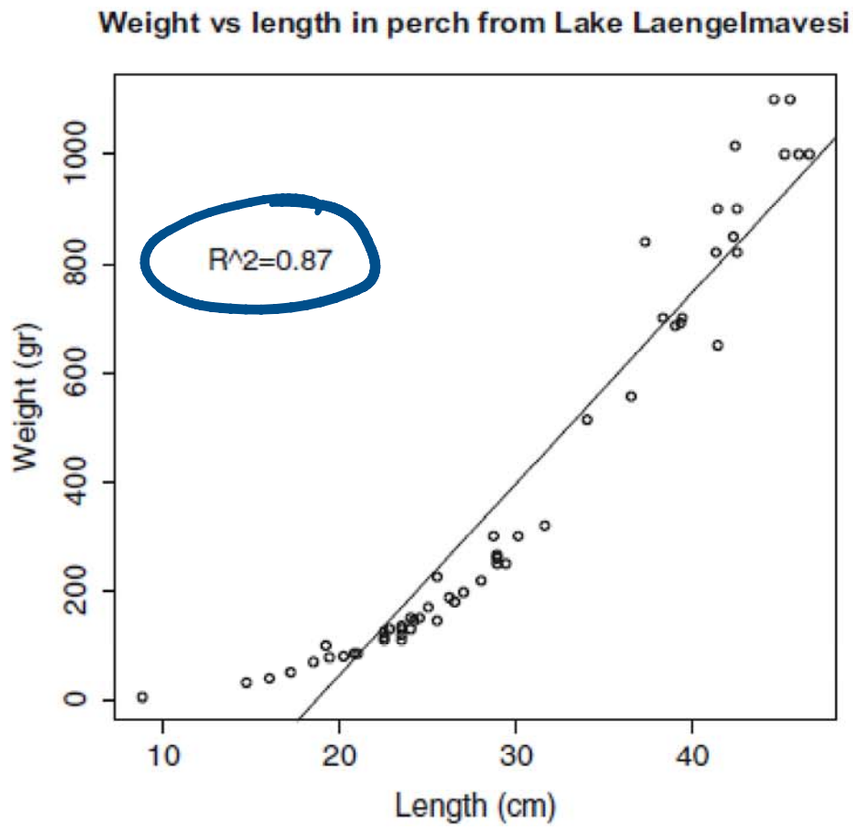
- (Ch 13) Regression
  - The regression problem
  - Training a linear regression model using least squares
  - Evaluating a model using the R-squared metric

# Today

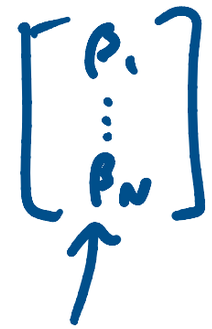
- (Ch 13) Regression
  - Outliers, overfitting and regularization
  - Nearest neighbors regression

# The regression problem

- Given a set of **feature vectors** $\mathbf{x}_i$ where each has a **numerical label** $y_i$, we want to train a model that can map unlabeled vectors to numerical values
- We can think of regression as fitting a line (or curve or hyperplane, etc.) to data
- Regression is like classification except that the prediction target is a number, not a class label (and that changes everything)



# Training a linear model



- Given a training dataset $\{(\mathbf{x}, y)\}$, we want to fit a model $y = \mathbf{x}^T\boldsymbol{\beta} + \xi$

- Define $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$ and $X = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}$ and $\mathbf{e} = \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_N \end{bmatrix}$

- To train the model, we must choose $\boldsymbol{\beta}$ that makes $\mathbf{e}$ small in the matrix equation

$$\mathbf{y} = X\boldsymbol{\beta} + \mathbf{e}$$

# Training using least squares

- In the least squares method, we aim to minimize $\|\mathbf{e}\|^2$

$$\|\mathbf{e}\|^2 = \|\mathbf{y} - X\boldsymbol{\beta}\|^2 = (\mathbf{y} - X\boldsymbol{\beta})^T(\mathbf{y} - X\boldsymbol{\beta})$$

- Differentiating and setting to zero (and skipping some matrix calculus) gives

$$X^T X\boldsymbol{\beta} - X^T\mathbf{y} = \mathbf{0}$$

- If $X^T X$ is invertible, the least squares estimate of the coefficients is

$$\widehat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T\mathbf{y}$$

# Training a linear model with constant offset

Model: $y = \beta_0 + \mathbf{x}^{(1)}\beta_1 + \mathbf{x}^{(2)}\beta_2 + \xi = \mathbf{x}^T\boldsymbol{\beta} + \xi$

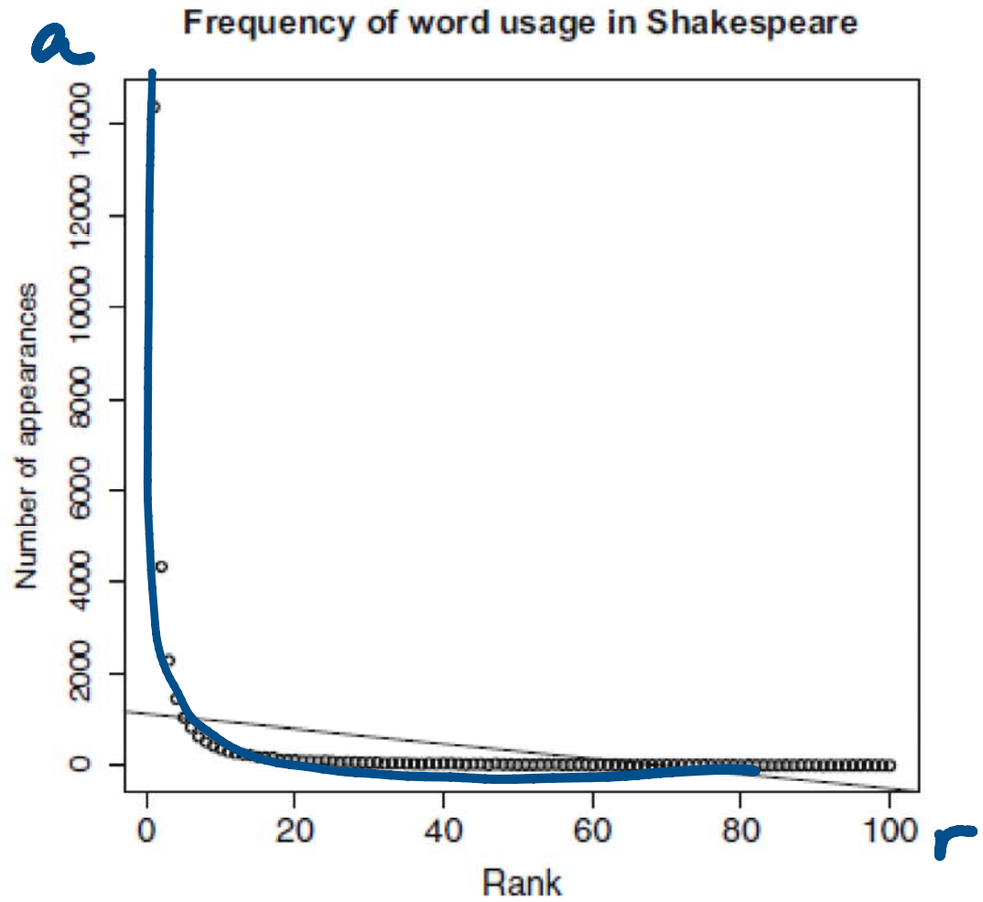
Training data

| 1 | $\mathbf{x}^{(1)}$ | $\mathbf{x}^{(2)}$ | $y$ |
|---|---|---|---|
| 1 | 1 | 3 | 0 |
| 1 | 2 | 3 | 2 |
| 1 | 3 | 6 | 5 |

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

# Dealing with nonlinear relationships

A linear model will not produce a good fit if the dependent variable is **not** linear in the explanatory variables



# Transforming variables to find a linear fit

In this example, taking natural log of both variables gives a linear fit

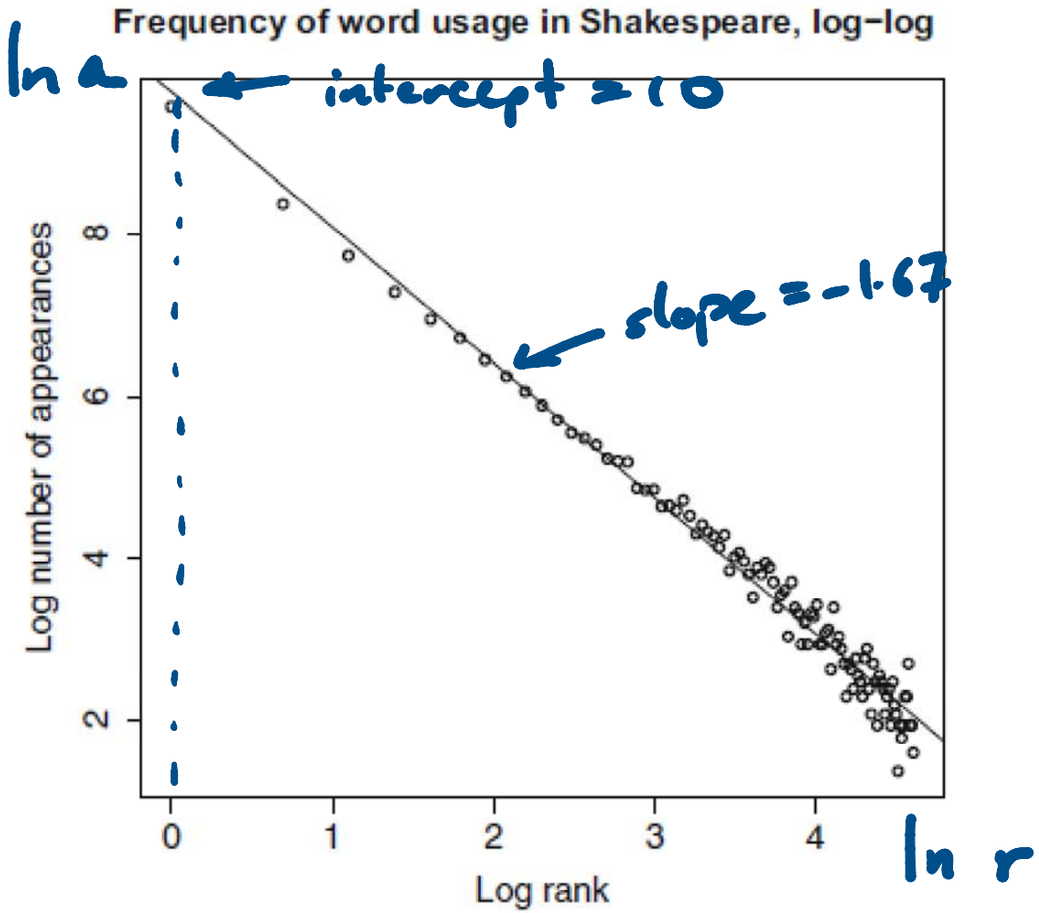
$$\ln a = -1.67 \ln r + 10$$

$$= \ln r^{-1.67} + 10$$

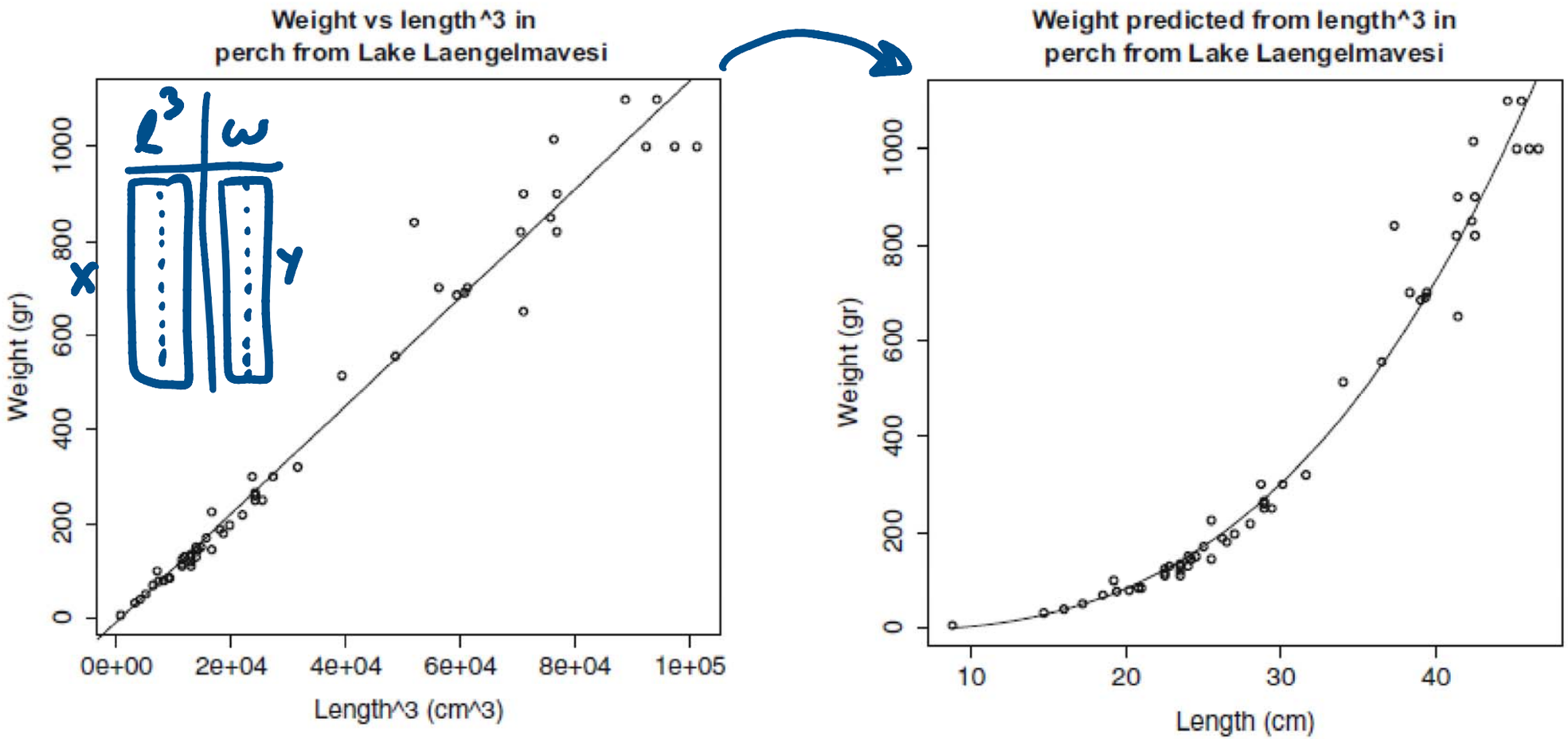
$$a = r^{-1.67}(e^{10})$$

$$a = e^{10}\left(\frac{1}{r}\right)^{1.67}$$

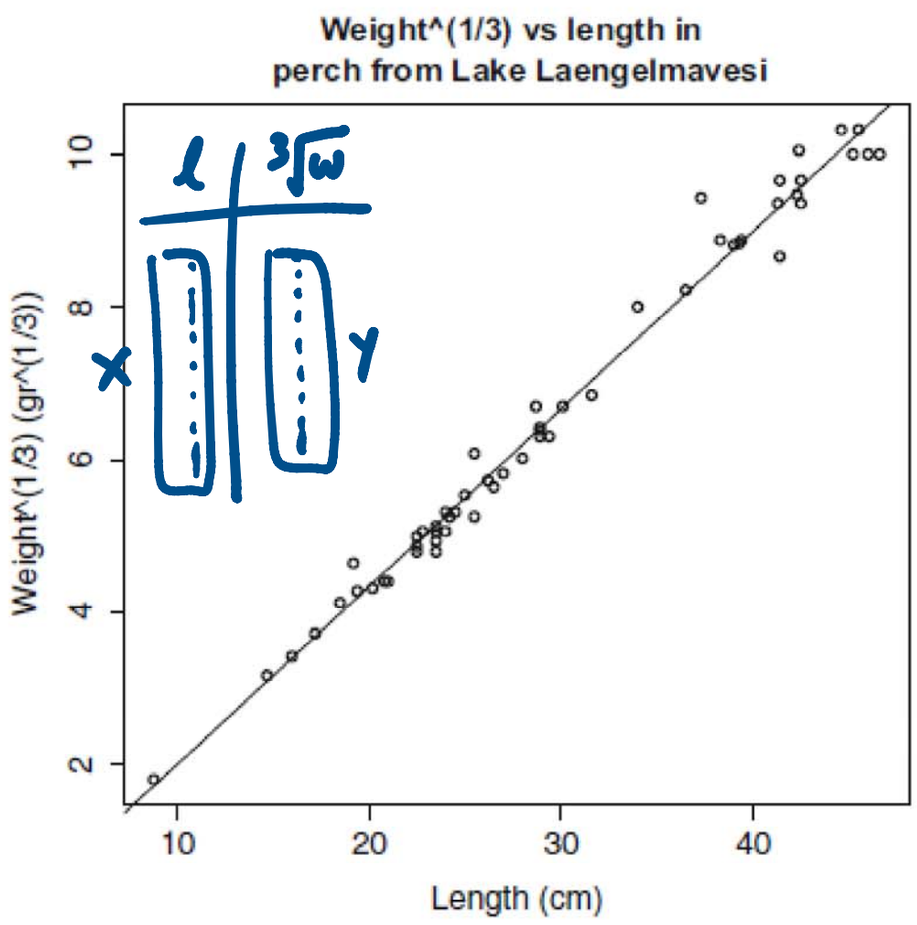
consistent with Zipf's law

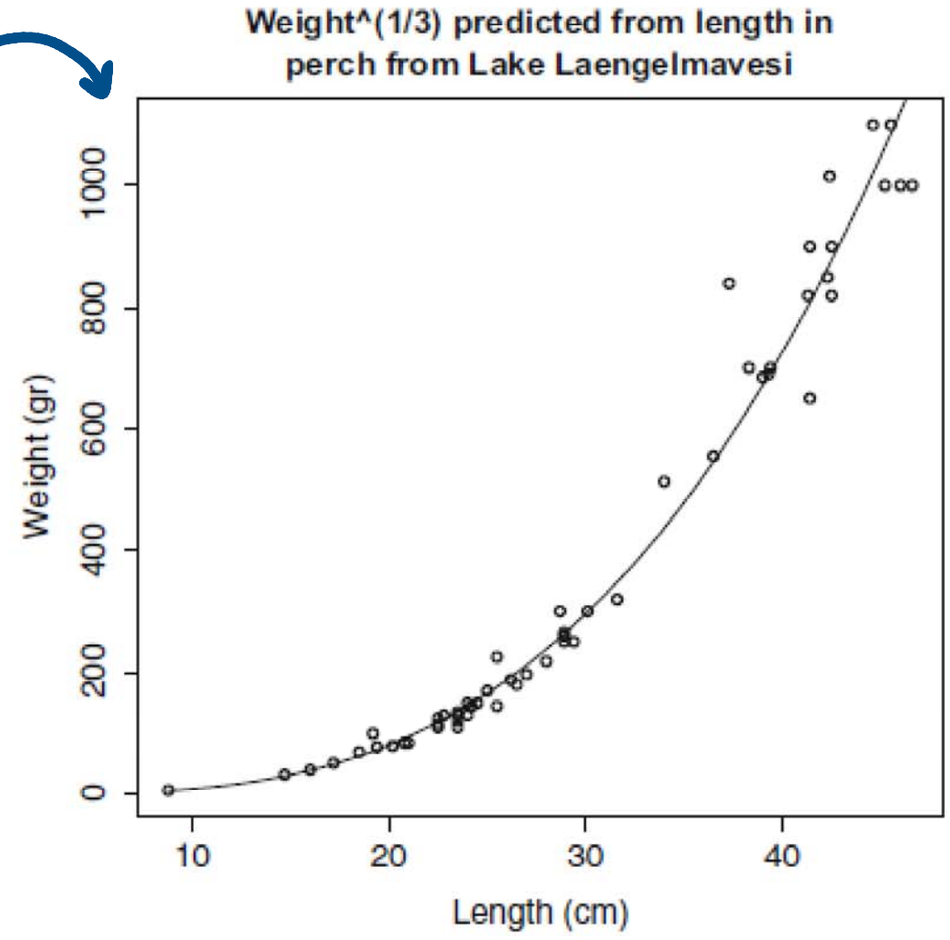


# Transforming just the explanatory variable



# Transforming just the dependent variable

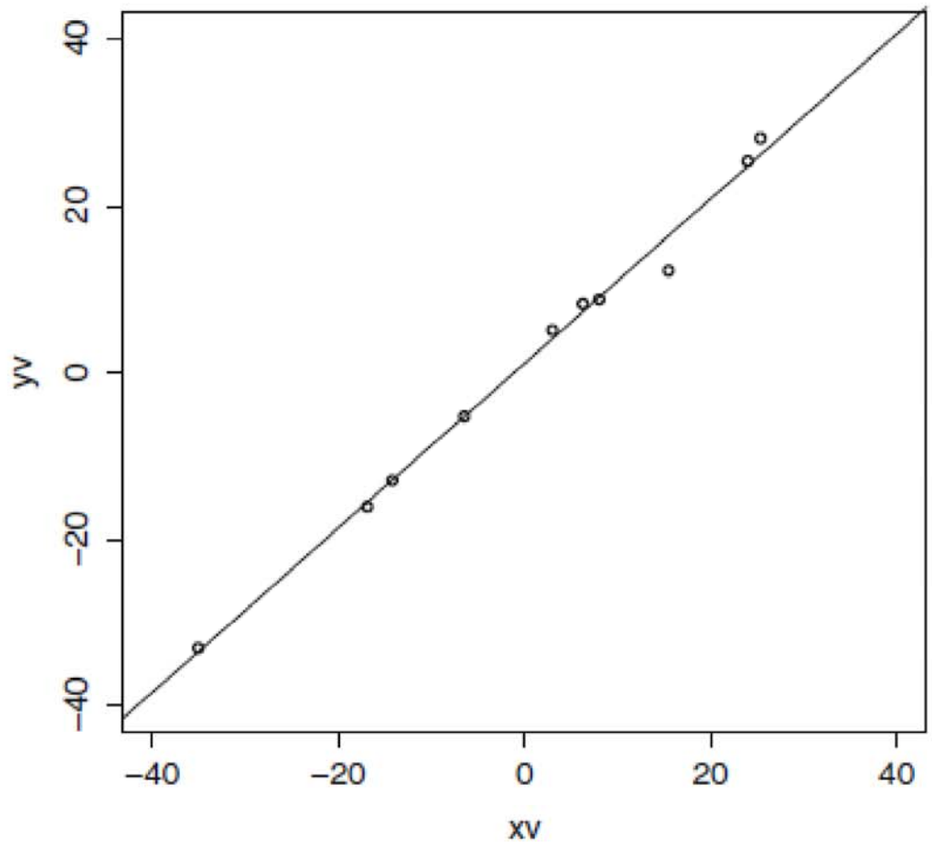


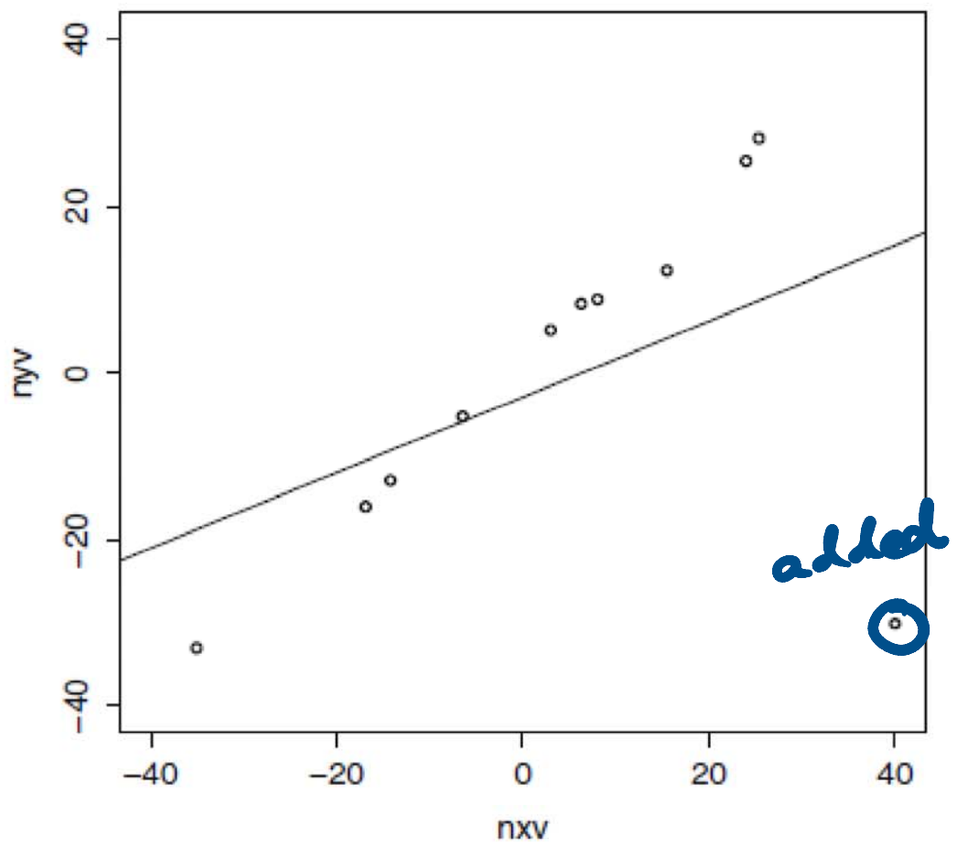



# Problems with the data

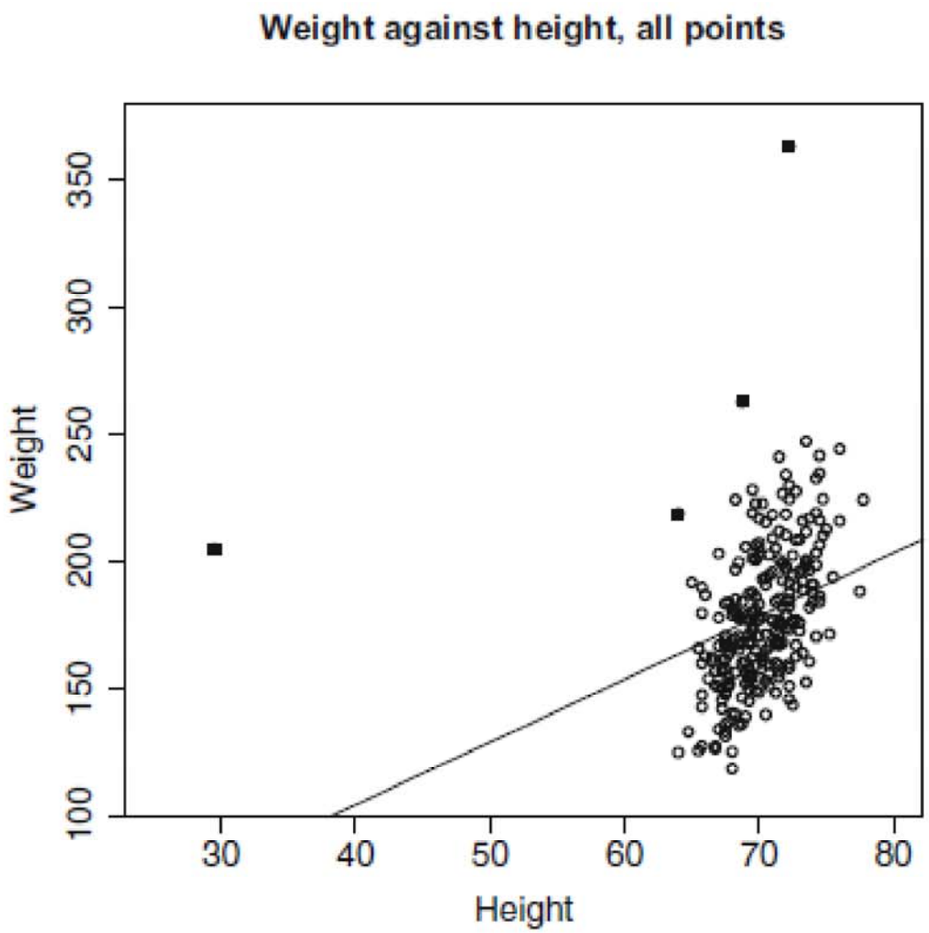
- Linear regression model parameters are very sensitive to outliers
- It is usually not obvious how to transform the explanatory variables
- Both of these problems can lead to **overfitting** the model

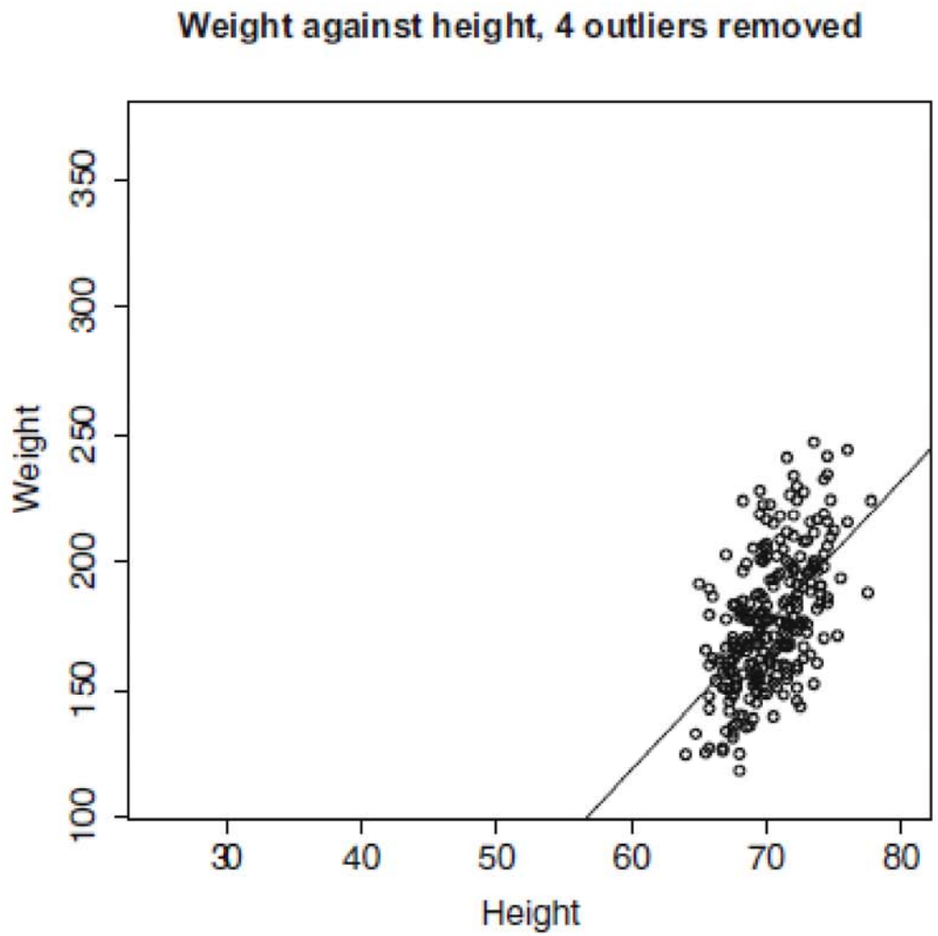
# Effect of outliers: synthetic data example



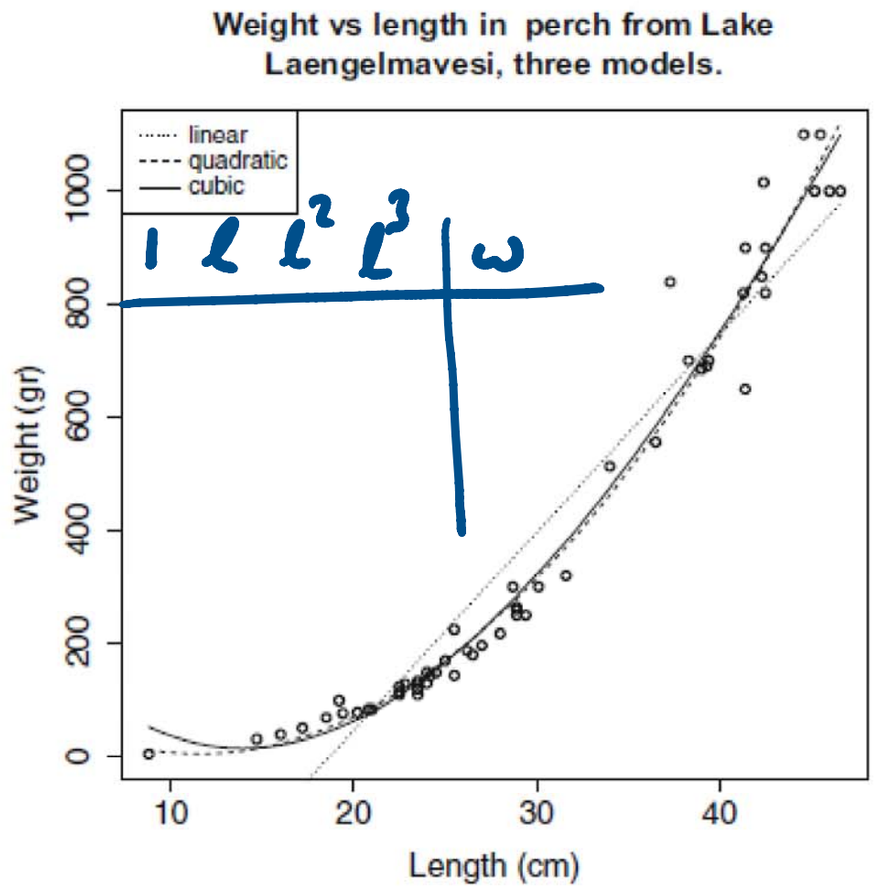



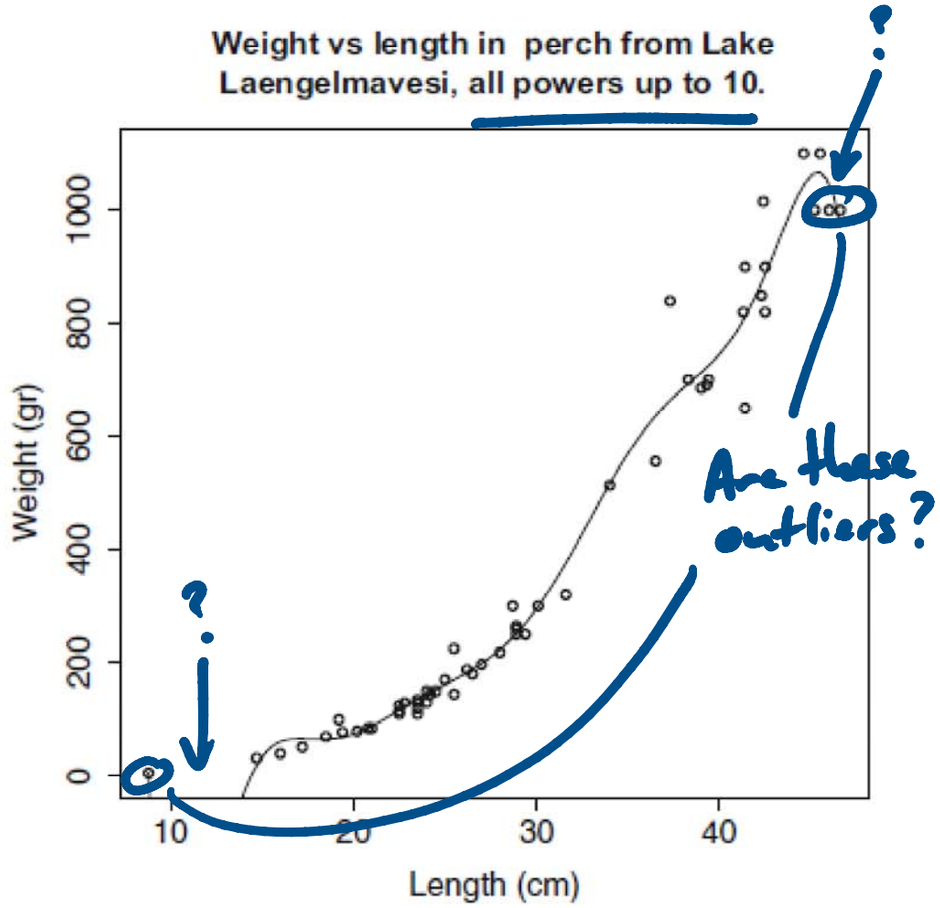
# Effect of outliers: body fat example



# Too many transformed explanatory variables



# Avoiding overfitting

- Method 1: validation
  - Use a validation set to choose the transformed explanatory variables
  - But the number of combinations is exponential in the number of variables
- Method 2: regularization
  - Impose a penalty on complexity of the model during the training
  - Less complex models have smaller model coefficients in the vector $\boldsymbol{\beta}$
- We can use validation to select the regularization parameter $\lambda$

# Regularizing the cost function

- In ordinary least squares, the cost function was $\|\mathbf{e}\|^2$

$$\|\mathbf{e}\|^2 = \|\mathbf{y} - X\boldsymbol{\beta}\|^2 = (\mathbf{y} - X\boldsymbol{\beta})^T(\mathbf{y} - X\boldsymbol{\beta})$$

- In regularized least squares, we add a complexity penalty weighted by $\lambda$

$$\|\mathbf{y} - X\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2 = (\mathbf{y} - X\boldsymbol{\beta})^T(\mathbf{y} - X\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

$\lambda$ sets the trade-off between the error term and the penalty

# Training using regularized least squares

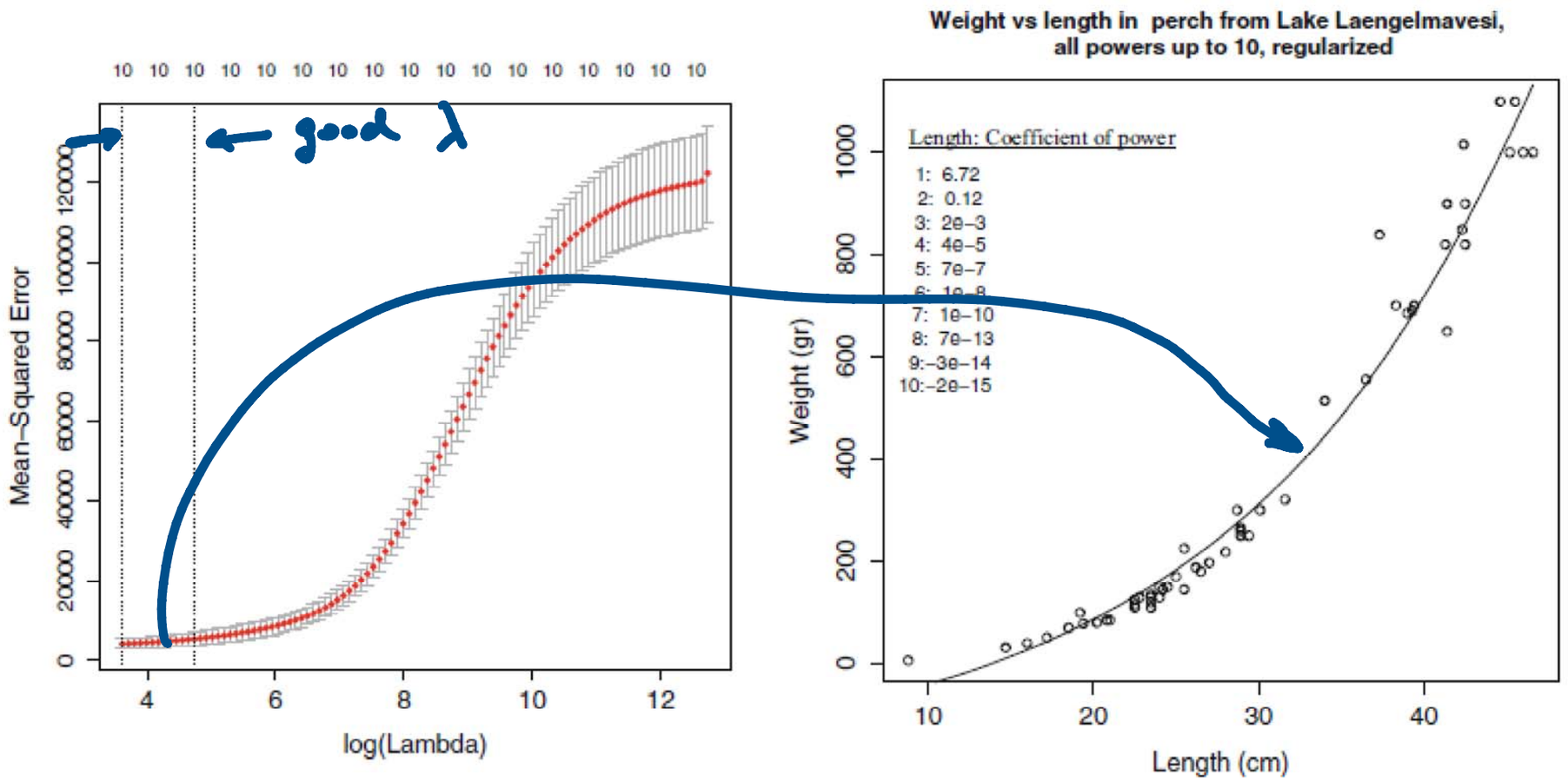
- Differentiating the cost function and setting to zero (and skipping some matrix calculus) gives

$$(X^T X + \lambda I)\boldsymbol{\beta} - X^T \mathbf{y} = \mathbf{0}$$

- $(X^T X + \lambda I)$ is always invertible, so the least squares estimate of the coefficients is

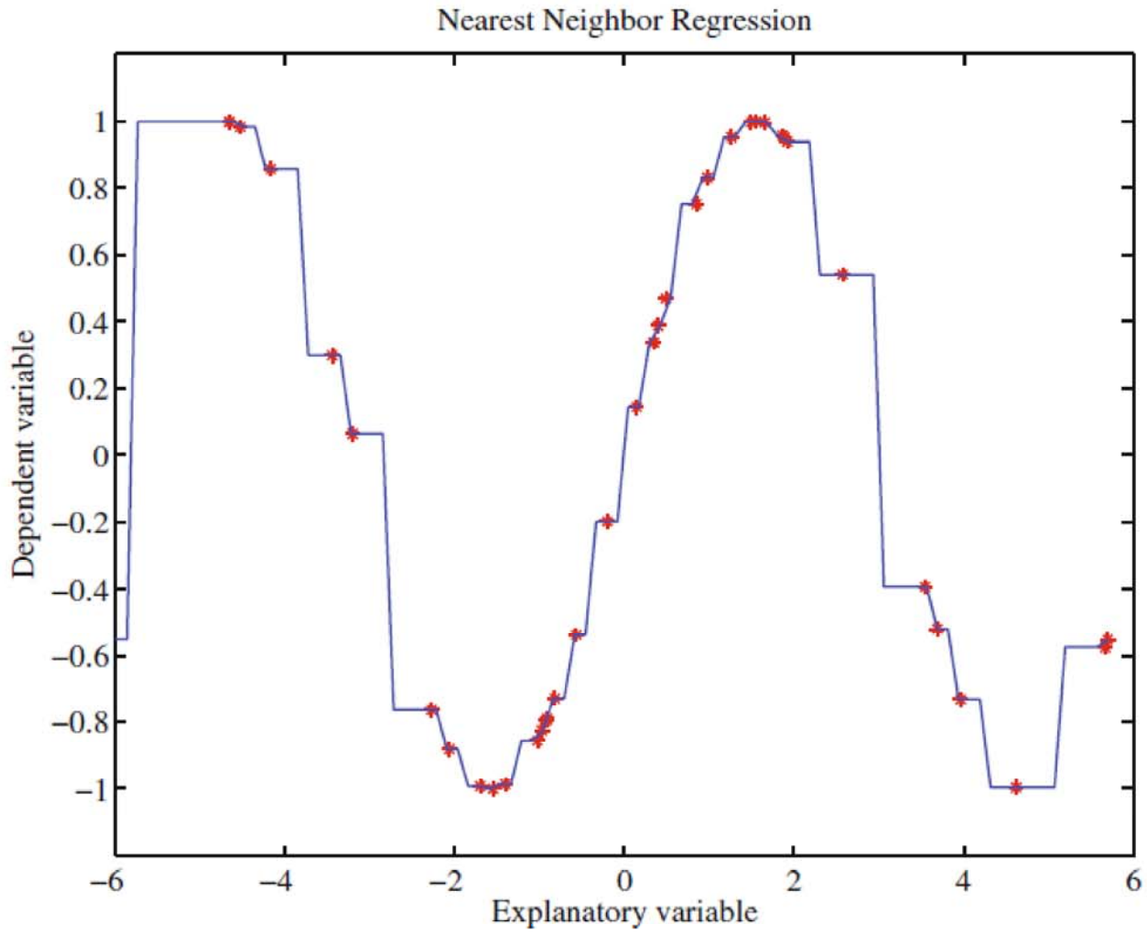
$$\widehat{\boldsymbol{\beta}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

# Choosing lambda using cross-validation tools



# Nearest neighbors regression

- A linear model is not the only solution to regression
- When there is plenty of data, $k$-nearest neighbors regression can be used
- $k = 1$ (shown on the right) is uncommon



# $k$-nearest neighbors with weights

The goal is to predict $\mathrm{y}_0^p$ from $\mathbf{x}_0$ from a training dataset $\{(\mathbf{x}, y)\}$

- Let $\{(\mathbf{x}_j, y_j)\}$ be the set of $k$ items such that $\mathbf{x}_j$ are nearest $\mathbf{x}_0$
- Predict

$$\mathrm{y}_0^p = \frac{\sum_j w_j y_j}{\sum_j w_j}$$

where $w_j$ are weights that drop off as $\mathbf{x}_j$ get further from $\mathbf{x}_0$

# 5-nearest neighbors with different weightings

- Inverse distance weighting

$$w_j = \frac{1}{\|\mathbf{x}_0 - \mathbf{x}_j\|}$$

- Exponential weighting

$$w_j = \exp\left(\frac{\|\mathbf{x}_0 - \mathbf{x}_j\|^2}{2\sigma}\right)$$